Python
字典 Dictionary

臺北科技大學資訊工程系

字典 dict

- 字典資料型別 {key1: value1, key2: value2, …}
  - 無順序性，不能像串列那樣切片。
  - key是 (immutable) 資料型別，value可以是任何資料型別。
  - 可用字典名稱[key]，「key」當作索引存取對應的value。
  - 修改字典中的value，使用字典名稱[key]=new Value。
  - 增加字典中新的key-value pair (字典是動態結構)
    - 字典名稱[new key]=new Value。
  - 使用del刪除key-value pair ，del 字典名稱[key]。
  - keys() 回傳字典所有的key，資料型別 (dict_keys)
  - values()回傳字典所有的value，資料型別 (dict_values)
  - items() 回傳字典所有的key-value(dict_items)，可用於for迴圈。
  - 利用in和not in運算子，可檢查字典中是否存在某個key/value。

Dictionary字典

- 建立字典

key　value

```
english = {'book':'書本', 'food':'食物' }  # key : value 配對的資料
book_c = english['book']                   # 使用key取得value
print(book_c)

d1 = {}                                    # 利用大括弧{ }建立字典
d2 = dict()
print(d1, d2)
```

```
書本
{} {}
```

Dictionary字典

- 字典動態結構

```
english = {'game':'遊戲', 'food':'食物'}

english['cat'] = '貓'            #增加新的 key-value pair
english['game'] = '比賽'         #修改字典中的value
del english['game']              #del刪除key-value pair

if 'game' in english:            #利用in, not in運算子檢查存在某key或value
    print(english['game'])
else:
    print('None')
```

```
None
```

Exercise

❑ 輸出

```
def f():
    data = {'name':'Tom', 'age':18, 'phone':'0933123001'}
    print(data)
    del data['name']                    #刪除
    print(data)
    studentAge = data['age']            #以 key 取值
    print(studentAge)
    if 'age' in data:                   #以 in 檢驗 key 是否存在
        print(data['age'])
```

```
x = ('key1', 'key2', 'key3')
y = 0
thisdict = dict.fromkeys(x, y)
print(thisdict)
```

```
{'key1': 0, 'key2': 0, 'key3': 0}
```

Dictionary字典

- 字典keys()、values()和items()方法
  - 回傳字典的keys，values或key-value pair的列表。
  - 料型別分別是dict_keys、dict_values和dict_items，可用於for。

```
d3 = { 'name':'John', 'age':20 }

for key in d3:
    print(key)

for key in d3.keys():
    print(key)

for value in d3.values():
    print(value)

for key, value in d3.items():
    print(key, value)
```

```
name
age
name
age
John
20
name John
age 20
```

Exercise

❑ 輸出

```
passwd={'Mars':0,'Mark':56}
passwd['Happy']=99
passwd['Smile']=12
del passwd['Mars']
passwd['Mark']=passwd['Mark']+1
print (passwd)
print (passwd.keys())
print (passwd.get('Tony'))
```

輸出

```
[redacted]7}
```

```
car = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
x = car.items()
print(x)
```

```
dict_items([([redacted])])
```

Exercise

❑ 輸出

```
car = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
x = car.values()
print(x)
y = car.keys()
print(y)
for key in car.keys():
    print(car[key])
```

```
dict_values(['Ford',                 ])
```

```
dict_keys(['brand',                 ])
```

Exercise

❑ 輸出?

```python
def tr(index):
    table = [chr(w) for w in range(ord('A'), ord('Z') + 1)]
    if int(index)>26: return '-1'
    return table[int(index)-1]

def check(x, data):
    if len(x)==0: return True
    if len(x)==1:
        data[x]= tr(x)
        return True
    if len(x)>=2:
        if tr(x[0:2])!='-1':
            data[x[0:2]]=tr(x[0:2])
            check(x[2:], data)
        check(x[:1], data)
        check(x[1:], data)
    return True
```

```python
data ={}
check('52', data)
print(data)

data ={}
check('25', data)
print(data)

data ={}
check('625', data)
print(data)
```

```
{'5': 'E', '2': 'B'}
{'
{'                                    }
```

Exercise

❑ 刪除

```
car = {'brand': 'Ford', 'model': 'Mustang',  'year': 1964}
car.popitem()
print(car)
car.pop('model')
print(car)
x = car.setdefault('color', 'White')
print(x)
print(car)
```

```
{'brand': 'Ford', 'model': 'Mustang'}
{'brand': 'Ford'}
White
{'brand': 'Ford', 'color': 'White'}
```

計算字母出現次數

可以改成計算 word 出現次數

- get
  - 回傳key的value
  - 如果沒有值，回傳第二個參數的值

```
s = 'I_am_a_programmer_and_I_have_no_life'
my_dict = {}
for letter in s:
    my_dict[letter] = my_dict.get(letter, 0) + 1
print(my_dict)
```

```
{'I': 2, '_': 8, 'a': 5, 'm': 3, 'p': 1, 'r': 3, 'o': 2, 'g': 1, 'e': 3, 'n': 2, 'd': 1, 'h': 1, 'v': 1, 'l': 1, 'i': 1, 'f': 1}
```

```
s = 'I_am_a_programmer_and_I_have_no_life'
charSet = set(s)
my_dict = {k:s.count(k) for  k in charSet}
print(sorted(my_dict.items(), key=lambda x: x[1]))
```

```
[('v', 1), ('l', 1), ('p', 1), ('f', 1), ('h', 1), ('g', 1), ('i', 1), ('d', 1), ('n', 2), ('I', 2), ('o', 2), ('r', 3), ('m', 3), ('e', 3), ('a', 5), ('_', 8)]
```

Dictionary轉型與合併

- dict()轉型與合併 update()

```
studentC = [('name','Mary'),('age',17)]
y=dict(studentC)
print(y)
studentD = (('English',80),('Chinese',85))
z=dict(studentD)
print(z)
y.update(z)
print(y)
y.clear()
print(y)
```

```
{'name': 'Mary', 'age': 17}
{'English': 80, 'Chinese': 85}
{'name': 'Mary', 'age': 17, 'English': 80, 'Chinese': 85
{}
```

Exercise

- 輸入一個學生2項資料(id, name), (age, address)存入2個list，再轉成2個字典，之後合併兩個字典，輸出合併的字典。

```
def ex():
    sid = input('id:')
    name = input('name:')
    s1 = [[sid, name]]
    d1 = dict(s1)
    age = input('age:')
    address = input('address:')
    s2 = [[[redacted]]]
    d2 = dict[redacted])
    d1.update([redacted])
    print(d1)
```

```
id:101

name:John

age:18

address:Taipei
{'101': 'John', '18': 'Taipei'}
```

Exercise

- 寫一個程式輸入n個學生姓名與成績，以字典儲存
  - 計算輸出所有學生成績的平均。
  - 輸入一個名字當key，從字典移除這個學生成績，把剩下的名字與成績輸出。

```
def ex(n):
    student = {}
    for i in range(n):
        name = input('name:')
        score = int(input('score:'))
        student[name]=score
    name = input('del name:')
    print('average=',sum(student.[redacted]))/n)
    del student[[redacted]]
    for name, score in student.[redacted]:
        print(name, score)

ex(2)
```

最大值

- 找出字典最大值與其Key

```
data = {'Tom': 90, 'John': 85, 'Mary': 75, 'Kevin': 90}
name = max(data, key = data.get)
print(name)
maxValue = max(data.values())
print(maxValue)
for keys, values in data.items():
    if values == maxValue:
        print(keys, values)
```

```
Tom
90
Tom 90
Kevin 90
```

Dictionary函式

Method	Description
clear()	清空
copy()	回傳副本
fromkeys()	Returns a dictionary with the specified keys and values
get()	Returns the value of the specified key
items()	Returns a list containing a tuple for each key value pair
keys()	Returns a list containing the dictionary's keys
pop()	Removes the element with the specified key
popitem()	Removes the last inserted key-value pair
setdefault()	Returns the value of the specified key. If the key does not exist: insert the key, with the specified value
update()	Updates the dictionary with the specified key-value pairs
values()	Returns a list of all the values in the dictionary

Exercise

- 取得 key, value 和個數

```
num = {1: 10, 2: 20, 3: 30, 4: 40, 5: 50, 6: 60}
print("key  value  count")
count=0
for key, value in num.items():
    count=count+1
    print(key,'  ',value,'    ', count)
```

```
key  value  count
1     10     1
2     20     2
3     30     3
4     40     4
5     50     5
6     60     6
```

- 產生字典並輸出，{x: x*x}, x=1~n)

```
def ex(n):
    d = dict()
    for x in range(1,n+1):
        d[x]=x*x
    print(d)

    m = dict([[x, x*x] for x in range(1, n+1)])
    print(m)
ex(5)
```

```
{1: 1, 2: 4, 3: 9, 4: 16, 5: 25}
{1: 1, 2: 4, 3: 9, 4: 16, 5: 25}
```

Exercise

- 輸入3個學生姓名與國英數3科成績，轉成字典 {name:[score1, score2, score3]}
  - 輸出國文及格學生姓名、成績，以姓名排序
  - 輸出所有學生成績，以國文成績排序，由大到小

```
def ex(n, m, data):
    student = {}
    for i in range(n):
        s = data[i].split()
        student[s[0]]=[int(s[i]) for i in range(1, m+1)]
    report1 = sorted(student.items(), key = lambda x:x[0])
    print(report1)
    print([[x[0], x[1]] for x in report1 if x[1][0]>=60])
    report2 = sorted(student.items(), key = lambda x:x[1][0])
    for x in report2:
        print(x[0], x[1])

data = ['Tom 60 80 70', 'Mary 90 70 60', 'Kan 50 100 80']
ex(3,3, data)
```

```
[('Kan', [50, 100, 80]), ('Mary', [90, 70,
60]), ('Tom', [60, 80, 70])]
[['Mary', [90, 70, 60]], ['Tom', [60, 80,
70]]]
Kan [50, 100, 80]
Tom [60, 80, 70]
Mary [90, 70, 60]
```

Dictionary排序

- 排序
  - 字典dict的keys()是iterable物件，iter可尋訪iterable物件

```
num = {'n2': [2, 3], 'n3': [5, 1], 'n1': [3, 2]}
it = iter(num)         # 使用尋訪物件 iter 遍訪 dict
print(next(it))        # 尋訪 第1個物件 n2
print(next(it))        # 尋訪 第1個物件 n3
print(sorted(num))  # 根據 key 排序
```

```
n2
n3
['n1', 'n2', 'n3']
```

  - 字典dict的values()若是sortable，可依此排序 values

```
num = {'n1': [2, 3, 1], 'n2': [5, 1, 2], 'n3': [3, 2, 4]}
sorted_dict = {x: sorted(y) for x, y in num.items()}
print(sorted_dict)
```

```
{'n2': [1, 2, 3], 'n3': [1, 2, 5], 'n1': [2, 3, 4]}
```

Dictionary排序

- 排序 – 字典dict的keys(), values(), items()是iterable物件，

```
num = {'n3': [2, 3, 1], 'n1': [5, 1, 2], 'n4': [3, 2, 4],'n2':[6, 1, 1]}
#字典排序，只取出key排序
sorted_key = sorted(num.keys(), reverse=True)
print(sorted_ key)
sorted_key = sorted(num)
print(sorted_ key)

#字典排序，根據value排序
#d.items() 將 d 轉換為iterable物件 ('n3', [2, 3, 1]), (...), (...)
# item[1] 表示 value, 根據 value 排序
sorted_dict = sorted(num.items(), key=lambda item:item[1])
print(sorted_dict)
#字典排序，根據key排序
# item[0] 表示 keys，根據 key 排序
sorted_dict = sorted(num.items(), key=lambda item:item[0])
print(sorted_dict)
```

```
['n4', 'n3', 'n2', 'n1']
['n1', 'n2', 'n3', 'n4']
```

```
[('n3', [2, 3, 1]), ('n4', [3, 2, 4]), ('n1', [5, 1, 2]), ('n2', [6, 1, 1])]
[('n1', [5, 1, 2]), ('n2', [6, 1, 1]), ('n3', [2, 3, 1]), ('n4', [3, 2, 4])]
```

Exercise

- 理想大學環境
  - 每一大學可用下列七種屬性表示：
    - BC(Big Campus)：代表有大校園。
    - NC(Next to City)：代表鄰近有大城市。
    - CT(Convenient Transportation)：代表交通方便。
    - NS(Next to Sea)：代表靠海。
    - NM(Next to Mountain)：代表依山。
    - HL(Has Lake)：代表校園有湖。
    - NL(Near Landscape)：代表附近有風景區。
  - 輸入說明：
    - 1. 輸入理想大學條件，用 + 號區格條件是 "或" 的關係，沒有 + 區隔是 "且" 的關係，屬性間和 +間以空白間隔。例如： BC NS + CT HL是找有大校園且靠海，或交通方便且校園有湖的所有大學。

Exercise

- 2. 第一行一正整數，代表大學個數 n (n<=10)。
- 3. 其後 n 行，每一行為大學名稱，接著大學具備屬性。大學名稱最多 10 個字母，屬性 2 個字母，均為英文字母，大學名稱及屬性間以一個空白分隔。
- 4. 接下來一行正整數m，為查詢個數， m<=10 。
- 5. 其後 m 行，每一行有一個查詢。條件為校園屬性。

輸出說明：(1, 2輸出各得 1/2分數)

- 1. m 行，第 i 列印出第 i 個查詢中，所有符合之大學名稱。
  - (1) 若有多個大學符合一個查詢，大學間以空白分隔。
  - (2) 每行查詢輸出順序，根據先後查詢條件符合的大學順序。
- 2. 如果都沒有完全符合，則輸出最多符合的大學。

Exercise

Sample Input	Sample Output
5 **NSYSU NC CT NS NM** **NTU BC NC CT NS** **NCCU BC NL HL** **Providence BC NC** **NTHU BC NS** **2** **BC NS + CT HL** **NM + BC NL + BC NC**	NTU NTHU NSYSU NCCU NTU Providence
5 **NSYSU NC CT NS NM** **NTU BC NC CT NS** **NCCU BC NL HL** **Providence BC NC** **NTHU BC NS** **1** **BC NS NL + CT HL**	NTU NTHU

Exercise

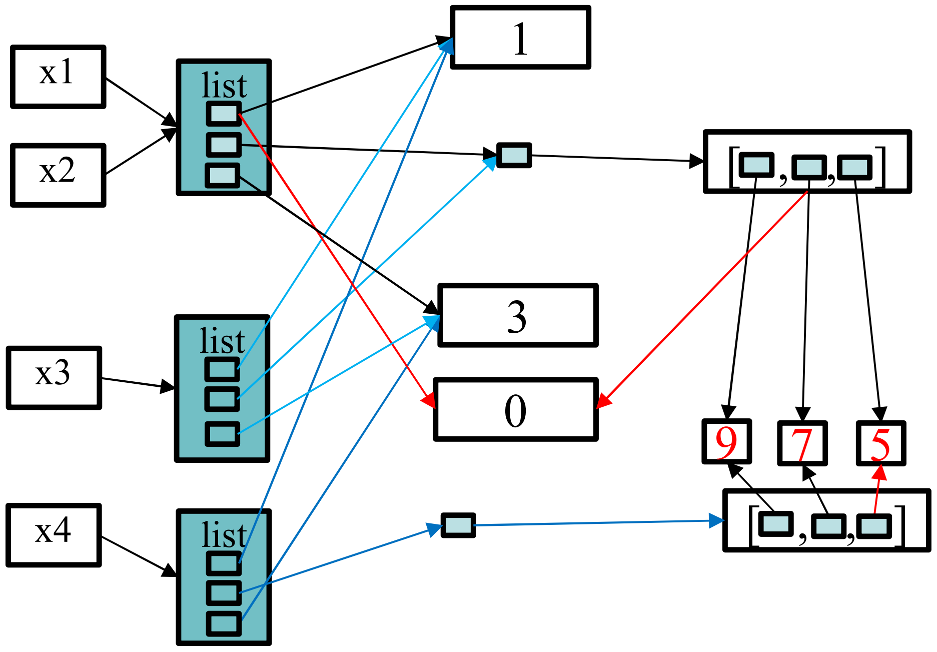
```
def getData():
    university = {}
    n = int(input())
    for i in range(n):
        item = input().split()
        university[item[0]] = set(i[redacted])
    return university

def match(con, feature):
    conds = con.split(' + ')
    maxNum = 0
    for i in range(len(conds)):
        if feature.issuperset(set(conds[i].split())):
            return -1
        k = len(feature&set([redacted]))
        if k>maxNum: maxNum = k
    return maxNum
```

```
def compute(con, university):
    data = {}
    for name, feature in university.items():
        data[name] = match(con, feature)
        if data[name] ==-1:
            print(name, end=' ')
    if -1 not in data.values():
        value = max(data.values())
        for key, v in data.items():
            if v==value: print(key, end=' ')
    print()

def main():
    university = getData()
    n = int(input())
    for i in range(n):
        con = input()
        compute(con, university)
```

Shallow Copy vs. Deep Copy

- **Assignment** (=) 造一個新變數**reference**指向原始字典物件
- 二種方式造字典:
  - **Shallow copy(淺拷貝)**:複製一層**references**
  - **Deep copy(深拷貝)**: 遞迴 (**recursively**) **複製** (**copies**) **所有層references**

```
import copy
x1 = [1, [9, 7, 5], 3]
x2 = x1
x3 = copy.copy(x1)
x4 = copy.deepcopy(x1)

x1[0] = 0
print(x1, x2)
print(x3, x4)
x1[1][1]=0
print(x1, x2)
print(x3, x4)
```

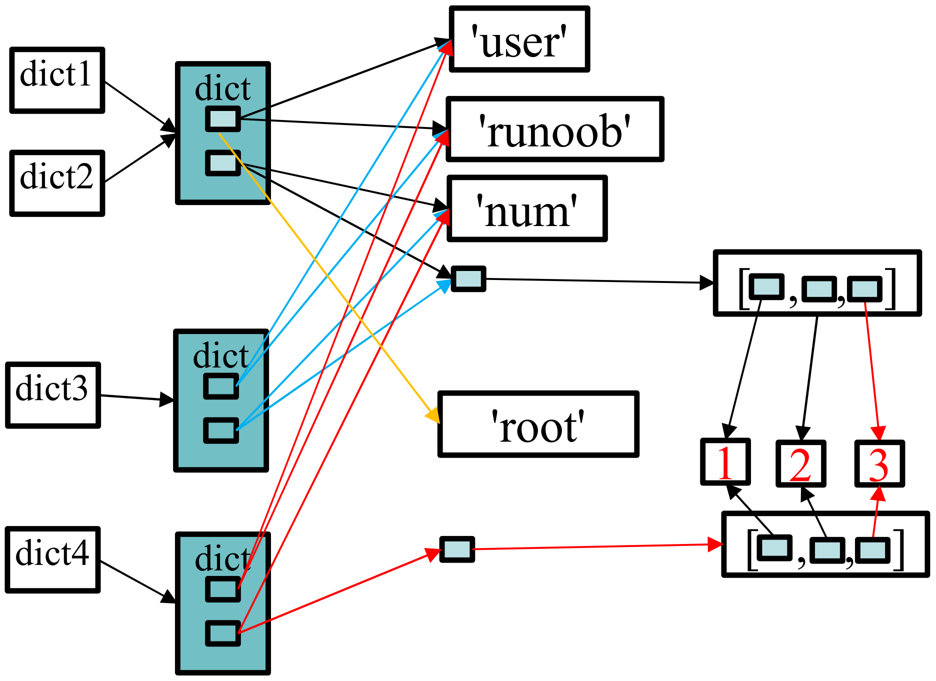


```
[0, [9, 7, 5], 3] [0, [9, 7, 5], 3]
[1, [9, 7, 5], 3] [1, [9, 7, 5], 3]
[0, [9, 0, 5], 3] [0, [9, 0, 5], 3]
[1, [9, 0, 5], 3] [1, [9, 7, 5], 3]
```

Shallow Copy vs. Deep Copy

```
import copy
dict1={'user':'runoob','num':[1,2,3]}
dict2=dict1
dict3=dict1.copy()
dict4=copy.deepcopy(dict1)
dict1['user']='root'
dict1['num'].remove(1)
print(dict1)
print(dict2)
print(dict3)
print(dict4)
del dict1['user']
print(dict1)
print(dict2)
print(dict3)
print(dict4)
```

```
{'user': 'root', 'num': [2, 3]}
{'user': 'root', 'num': [2, 3]}
{'user': 'runoob', 'num': [2, 3]}
{'user': 'runoob', 'num': [1, 2, 3]}
{'num': [2, 3]}
{'num': [2, 3]}
{'user': 'runoob', 'num': [2, 3]}
{'user': 'runoob', 'num': [1, 2, 3]}
```



巢狀字典

- 字典內包含字典稱巢狀字典(**nested dictionaries**)

```
myfamily = {
  "child1" : {
    "name" : "Emil",
    "year" : 2004
  },
  "child2" : {
    "name" : "Tobias",
    "year" : 2007
  },
  "child3" : {
    "name" : "Linus",
    "year" : 2011
  }
}
```

```
child1 = {
  "name" : "Emil",
  "year" : 2004
}
child2 = {
  "name" : "Tobias",
  "year" : 2007
}
child3 = {
  "name" : "Linus",
  "year" : 2011
}

myfamily = {
  "child1" : child1,
  "child2" : child2,
  "child3" : child3
}
```

巢狀字典

```
people = {1: {'name': 'John', 'age': '27', 'sex': 'Male'},
          2: {'name': 'Marie', 'age': '22', 'sex': 'Female'}}

print(people[1]['name'])
print(people[1]['age'])
print(people[1]['sex'])

people[3] = {}
people[3]['name'] = 'Luna'
people[3]['age'] = '24'
people[3]['sex'] = 'Female'
people[3]['married'] = 'No'
print(people[3])
people[4] = {'name': 'Peter', 'age': '29', 'sex': 'Male', 'married': 'Yes'}
print(people[4])

del people[3]['married']
del people[4]['married']
print(people[3])
print(people[4])
print()
del people[3], people[4]
print(people)
```

```
John
27
Male
{'name': 'Luna', 'age': '24', 'sex': 'Female', 'married': 'No'}
{'name': 'Peter', 'age': '29', 'sex': 'Male', 'married': 'Yes'}
{'name': 'Luna', 'age': '24', 'sex': 'Female'}
{'name': 'Peter', 'age': '29', 'sex': 'Male'}

{1: {'name': 'John', 'age': '27', 'sex': 'Male'}, 2: {'name':
'Marie', 'age': '22', 'sex': 'Female'}}
```

巢狀字典的尋訪

```
people = {1: {'Name': 'John', 'Age': '27', 'Sex': 'Male'},
          2: {'Name': 'Marie', 'Age': '22', 'Sex': 'Female'}}

for p_id, p_info in people.items():
    print("\nPerson ID:", p_id)

    for key in p_info:
        print(key + ':', p_info[key])
```

```
Person ID: 1
Name: John
Age: 27
Sex: Male

Person ID: 2
Name: Marie
Age: 22
Sex: Female
```

字典Comprehension

- 造字典，{數字:平方}

```
D = {}
D[0] = 0
D[1] = 1
D[2] = 4
D[3] = 9
D[4] = 16
print(D)    # {0: 0, 1: 1, 2: 4, 3: 9, 4: 16}
```

- 使用迴圈

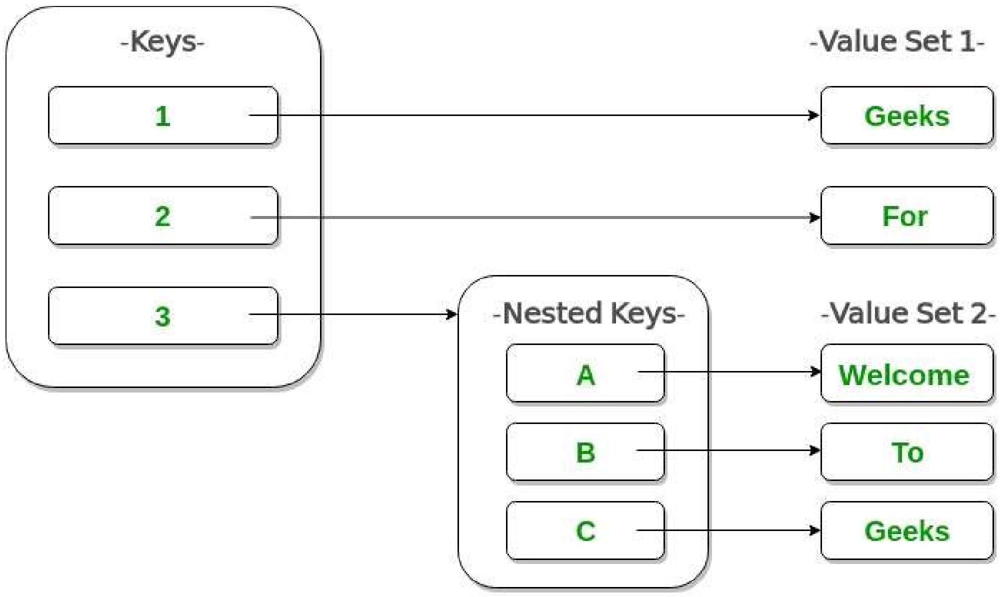
```
D = {}
for x in range(5):
    D[x] = x**2
print(D)    # {0: 0, 1: 1, 2: 4, 3: 9, 4: 16}
```

- 使用comprehension (類似串列產生器)

```
D = {x: x**2 for x in range(5)}
print(D)    # {0: 0, 1: 1, 2: 4, 3: 9, 4: 16}
```

Exercise

- 造巢狀字典



Letter Grade	Number Grade
A+	97
A	95
A-	92
B+	87
B	85
B-	82
C+	77
C	75
C-	72
D+	67
D	65
D-	62
F	50

```
Dict = {1: 'Geeks', 2: 'For', 3: {'A' : 'Welcome',                '}}
print(Dict)
```

學生成績

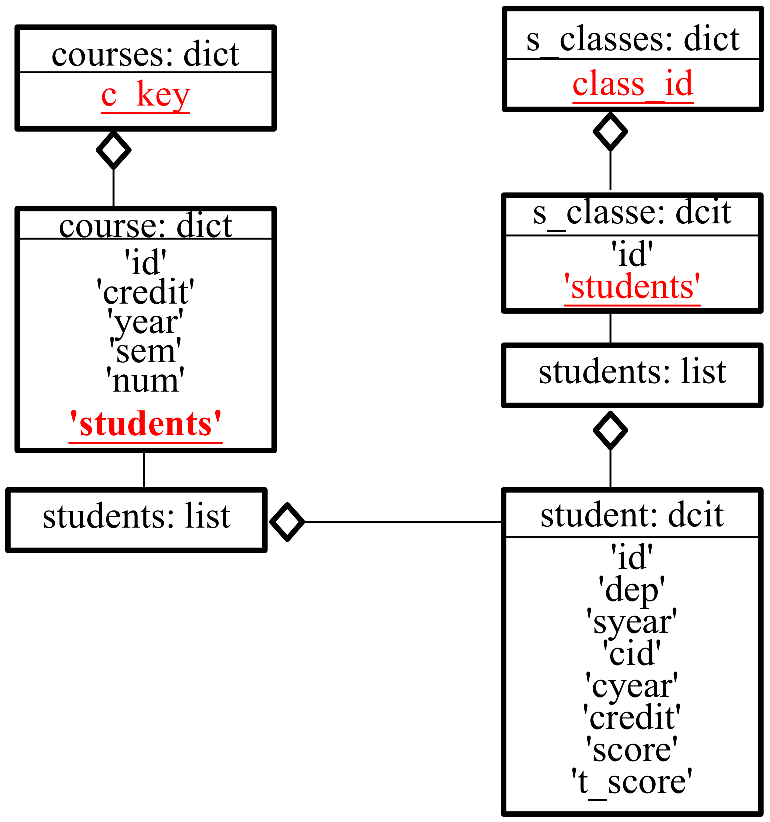
- 某學校有一系統的資料庫儲存各項課程資訊：
  - 輸出各科系、各年級的每一學年平均成績前三名的名次百分比和撤選百分比
  - 輸出每項課程每年的最高課堂成績、平均課堂成績、最低課堂成績、課堂成績標準差和撤選百分比、前三名的成績和名次百分比
  - 搜尋非國文一或英文一某項學科歷年課堂分數最高的前兩名學號和最多人數的科系
  - 搜尋國文一或英文一歷年會考分數最高的前兩名學號和最多人數的科系
  - 搜尋某一年的國文一或英文一會考分數最高的前兩名學號
- 每個課程含課程編碼、學期編碼、學生學校學號編碼和學生成績：
  - 學校學號：共八碼，前三碼是入學年度，中間三碼是科系編碼、後兩碼座號。
  - 課程編碼：共四碼，前三碼是學科編碼(整數，英文一是101、國文一是201)，後一碼是學分數(整數)。
  - 學期編碼：共四碼，前三碼是學年度(整數)、後一碼是學期(第一學期1、第二學期2）

學生成績

範例輸入說明：
2 （課程數）
1013 1121 9（開始輸入學科編碼101、學分數3、112學年度上學期的學生資訊）、9位學生
11153001 59 63（學生入學年度111年，科系530，號碼01，會考成績59，課堂成績63）
11153002 68 72（學生入學年度111年，科系530，號碼02，會考成績68，課堂成績62）
11153003 39 64（學生入學年度111年，科系530，號碼03，會考成績39，課堂成績64）
11153004 w （學生入學年度111年，科系530，號碼04，退選）
11153005 68 82（學生入學年度111年，科系530，號碼05，會考成績68，課堂成績82）
11253001 67 91（學生入學年度112年，科系530，號碼01，會考成績67，課堂成績91）
11253002 78 92（學生入學年度112年，科系530，號碼02，會考成績78，課堂成績92）
11253003 72 68（學生入學年度112年，科系530，號碼03，會考成績72，課堂成績68）
11253004 91 76（學生入學年度112年，科系530，號碼04，會考成績91，課堂成績76）
1214 1131 8（開始輸入學科編碼121、學分數4、113學年度上學期的學生資訊、8位學生）
11153001 63（學生入學年度111年，科系530，號碼01，課堂成績63）
11153002 75（學生入學年度111年，科系530，號碼02，課堂成績75）
11153003 82（學生入學年度111年，科系530，號碼03，課堂成績82）
11153004 92（學生入學年度111年，科系530，號碼04，課堂成績92）
11153101 82（學生入學年度111年，科系531，號碼01，課堂成績82）
11153102 79（學生入學年度111年，科系531，號碼02，課堂成績79）
11153103 90（學生入學年度111年，科系531，號碼03，課堂成績90）
11153104 70（學生入學年度111年，科系531，號碼04，課堂成績70）

學生成績

```
3
1012 1101 6
11029301 72 63
11029302 64 72
11029303 65 75
11029307 72 81
11029310 86 77
11059311 83 79
1021 1112 9
11029301 92
11059308 95
11029309 82
11029314 87
11059315 85
11059311 73
11029318 w
11029319 w
11029321 74
3333 1112 2
11029301 92
11059311 79
```



學生成績

```
#設定某一系級，學生修課成績
def setClass(s_classes:dict, student:dict)->None:
    s_class = s_classes.get(student['id'][:6],0)
    if s_class==0:
        s_class = dict()
        s_class['id']=student['id'][:6]
        s_class['students']=[student]
        s_classes[student['id'][:6]] = s_class
    else:
        s_class['students'].append(student)

#針對 N 門課程，輸入學生修課成績
def getData()->(dict, dict):
    courseNum = int(input())
    s_classes=dict()
    courses = dict()
    for i in range(courseNum):
        getCourseData(courses, s_classes)
    return s_classes, courses

#課程 key (課程編號 + 開課學年) 所有學生成績
def print_course(c_key:str, courses: dict)->None:
    print(courses[c_key]['id'])
    for student in courses[c_key]['students']:
        print(student['id'],student.get('score','W'))
```

```
#針對某一課程 ，輸入學生修課成績
def getCourseData(courses: dict, s_classes: dict)->None:
    data = input().split()
    cid = data[0]
    c_key = data[0] + data[1]
    credit = int(data[0][-1])
    c_year = int(data[1][:3])
    sem  = int(data[1][-1])
    num  = int(data[2])
    course = {'id':cid, 'credit':credit, 'year':c_year, 'sem':sem, 'num':num}
    students=[]
    for i in range(num):
        data = input().split()
        sid =  data[0]
        dep =  data[0][3:6]
        s_year = int(data[0][:3])
        student={'id':sid, 'dep':dep, 'syear':s_year, 'cid': cid, 'cyear':c_year,'credit':credit}
        student['drop'] = 'N'
        if data[1] =='w':
            student['drop'] = 'w'
        else:
            student['score']=int(data[1])
            if sid[:3]=='101' or sid[:3]=='201':
                student['t_score']=int(data[2])
        setClass(s_classes, student)
        students.append(student)
    course['students'] = students
    courses[c_key] = course
```

學生成績

```
# 科系年級 class_id, 學年 cyear, 所有學生修課資料
def print_class(s_classes: dict, class_id: str, cyear: int)->None:
    print('c_year=',cyear)
    for s in s_classes[class_id]['students']:
        if s['cyear']==cyear:
            print(s)

# 科系年級 class_id, 學年 cyear, 所有學生平均成績
def compute_class_student_avg(s_classes: dict, class_id: str, cyear: int)->None:
    print('avg c_year=',cyear)
    sid_set = {s['id'] for s in s_classes[class_id]['students'] if s['cyear']==cyear}
    data=[]
    print(sid_set)
    for sid in sid_set:
        credit, sum_score = 0, 0
        for student in s_classes[class_id]['students']:
            if student['id']==sid and student['cyear']==cyear:
                credit += student['credit']
                sum_score += student['score']*student['credit']
        if credit>0:
            #print(sid, credit, sum_score, sum_score//credit)
            data.append([sid, credit, sum_score, sum_score//credit])
    newData =sorted(data, key=lambda x: x[3], reverse=True)
    for d in newData:
        print(d[0], d[1],d[2], d[3])

s_classes, courses= getData()
print_course('10211112',courses)
print_course('10121101',courses)
print_class(s_classes, '110593', 110)
print_class(s_classes, '110593', 111)
compute_class_student_avg(s_classes, '110593', 111)
```

學生成績

```
1021
11029301 92
11059308 95
11029309 82
11029314 87
11059315 85
11059311 73
11029318 W
11029319 W
11029321 74
1012
11029301 72
11029302 64
11029303 65
11029307 72
11029310 86
11059311 83
c_year= 110
{'id': '11059311', 'dep': '593', 'syear': 110, 'cid': '1012', 'cyear': 110,
'credit': 2, 'drop': 'N', 'score': 83}
c_year= 111
{'id': '11059308', 'dep': '593', 'syear': 110, 'cid': '1021', 'cyear': 111,
'credit': 1, 'drop': 'N', 'score': 95}
{'id': '11059315', 'dep': '593', 'syear': 110, 'cid': '1021', 'cyear': 111,
'credit': 1, 'drop': 'N', 'score': 85}
{'id': '11059311', 'dep': '593', 'syear': 110, 'cid': '1021', 'cyear': 111,
'credit': 1, 'drop': 'N', 'score': 73}
{'id': '11059311', 'dep': '593', 'syear': 110, 'cid': '3333', 'cyear': 111,
'credit': 3, 'drop': 'N', 'score': 79}
avg c_year= 111
{'11059308', '11059315', '11059311'}
11059308 1 95 95
11059315 1 85 85
11059311 4 310 77
```

END