程式設計

1082數位教材

單元4:資料型態

主講老師：徐培倫

單元4 資料型態

- 資料型態(Data Type)
- 有序序列
  - 串列(list) ：使用[ ]
  - 序對(tuple) : 使用( )
- 無序序列
  - 集合(set) ：使用 { }
  - 字典(dict) :使用 { }

資料型態(Data Type)

- 數值型(numeric) 和 序列型(sequence)

Data type		name	Order	mutable	Example
整數	numeric	int	N/A	immutable	10
浮點數		float	N/A	immutable	3.14
複數		complex	N/A	immutable	2+4j
字串	sequence	str	N/A	immutable	"Hello"
字節		bytes	N/A	immutable	
字節陣列		bytearry	N/A	mutable	
串列		list	有序	mutable	[1,"Two",3+3j]
序對		tuple	有序	immutable	(3,4,[6,8,10])
集合		set	無序	mutable	{5,3,2,1}
字典		dict	無序	key:immutable	{"A":10, "B":20}

資料型態(Data Type)

name	Example
int	10
float	3.14
complex	2+4j
str	"Hello"
bytes	
bytearry	
list	[1,"Two",3+3j]
tuple	(3,4,[6,8,10])
set	{5,3,2,1}
dict	{"A":10, "B":20}

```
>>> print (type(3.14))
<class 'float'>
>>> print (type(2+4j))
<class 'complex'>
>>> print (type("Hello"))
<class 'str'>
>>> print (type([1,2,3,"Two"]))
<class 'list'>
>>> print (type({1,2,3}))
<class 'set'>
>>> print (type((1,2,3)))
<class 'tuple'>
>>> print (type( {"a":1, "b":2}))
<class 'dict'>
>>>
```

序列型資料可進行的運算

Operation	Result
x in s	x 是否在 s 中
x not in s	x 是否不在 s 中
s + t	結合 s 和 t
s * n OR n * s	s 重複 n 次連結 s
s[i]	取出索引值 i 的元素
s[i : j]	取出索引值 i 到 j-1 的元素 (不包含 j)
s[i : j : k]	取出索引值 i 到 j-1 的元素, 間隔 k
len(s)	回傳 s 的長度
min(s)	回傳 s 的最小值
max(s)	回傳 s 的最大值
s.index(x)	回傳 s 中第一次出現 x 的索引值
s.count(x)	回傳 s 中 x 出現的次數

序列型運算範例(4-1oper.py)

```
 8 s=[1,2,3,4,5,6,7,8]
 9 t=['a','b','c']
10 x=3
11 print ("s=",s)
12 print ("x=",x)
13 print ("x in s =", x in s)
14 print ("x not in s =", x not in s)
15 print ("t=",t)
16 print ("s+t=",s+t)
17 print ("s*2=",s*2)
18 print ("3*t=",3*t)
19 print ("s[3]=",s[3]) #index 由 0 開始
20 print ("S[1:5]=",s[1:5]) # 不含 s[5], 1-4
21 print ("S[1:5:2]=",s[1:5:2])
22 print ("len,min,max=",len(s),min(s),max(s))
23 print ("s.index(5)=",s.index(5))
24 print ("s.count(5)=",s.count(5))
```

```
s= [1, 2, 3, 4, 5, 6, 7, 8]
x= 3
x in s = True
x not in s = False
t= ['a', 'b', 'c']
s+t= [1, 2, 3, 4, 5, 6, 7, 8, 'a', 'b', 'c']
s*2= [1, 2, 3, 4, 5, 6, 7, 8, 1, 2, 3, 4, 5, 6, 7, 8]
3*t= ['a', 'b', 'c', 'a', 'b', 'c', 'a', 'b', 'c']
s[3]= 4
S[1:5]= [2, 3, 4, 5]
S[1:5:2]= [2, 4]
len,min,max= 8 1 8
s.index(5)= 4
s.count(5)= 1
```

作業4-1

- 請寫出運算結果

s=[5,7,6,8,4]

t=[1,3,2,3]

根據 s+2*t 的串列,請簡答下列問題

(1) print (s+2*t)

(2) 最小值=

(3) 最大值=

(4) 長度 =

(5) 加總 =

(6) 排序 =

(7) t.count(3) =

(8) t.index(3) =

串列(list)：使用中括號[ ]

- 列表是Python中最具靈活性的有序集合物件類型,
- 與字串不同的是,列表可以包含任何種類的物件:
- 數字,字串,甚至是其他列表.
- 並且列表都是可變物件

LIST 列表有一些內置的函數對列表進行增加,修改和刪除等操作

- append(x) 在列表尾部追加單個物件x。使用多個參數會引起異常。
- count(x) 返回物件x在列表中出現的次數。
- extend(L) 將列表L中的表項增加到列表中。返回None。
- Index(x) 返回列表中匹配物件x的第一個列表項的索引。無匹配元素時產生異常。
- insert(i,x) 在索引為i的元素前插入物件x。如list.insert(0,x)在第一項前插入物件。返回None。
- pop(x) 刪除列表中索引為x的表項，並返回該表項的值。若未指定索引，pop返回列表最後一項。
- remove(x) 刪除列表中匹配物件x的第一個元素。匹配元素時產生異常。返回None。
- reverse() 顛倒列表元素的順序。
- sort() 對列表排序，返回none。
- max(), min() 最大最小

LIST 範例

- #定義一個列表
- listA = ['a', 'b', 'c', 1, 2]
- print(listA)
- # 向 list 中增加元素
- # 1.使用append 向 list 的末尾追加單個元素。
- listA.append(3)
- print (listA)
- # 2.使用 insert 將單個元素插入到 list 中。數值參數是插入點的索引
- listA.insert(3, 'd') # 在下標為3處增加一個元素
- print (listA)
- # 3.使用 extend 用來連接 list
- listA.extend([7, 8])
- print (listA)

```
['a', 'b', 'c', 1, 2]
['a', 'b', 'c', 1, 2, 3]
['a', 'b', 'c', 'd', 1, 2, 3]
['a', 'b', 'c', 'd', 1, 2, 3, 7, 8]
```

Append vs Extend

- # extend 和 append 看起來類似，但實際上完全不同。
- # extend 接受一個參數，這個參數總是一個 list，
- # 並且把這個 list 中的每個元素增加到原 list 中。
- # 另一方面，append 接受一個參數，這個參數可以是任何資料類型，並且簡單地追加到 list 的尾部。
- listA.append(['a',100])
- print (listA)

```
['a', 'b', 'c', 'd', 1, 2, 3, 7, 8, ['a', 100]]
10
6
0
False
```

- # 獲取列表的長度
- print (len(listA)) # 10

- # 在 list 中搜索
- print (listA.index(3)) # index 在 list 中查找一個值的首次出現並返回索引值。
- print (listA.index('a')) # 如果在 list 中沒有找到值，Python 會引發一個異常。
- print (5 in listA) # 要測試一個值是否在 list 內，使用 in。如果值存在，它返回 True，否則返為 False 。

從 list中刪除元素

- # 從 list 中刪除元素
- listA.remove(3) # remove 從 list 中 僅僅 刪除一個值的首次出現。如果在 list 中沒有找到值，Python 會引發一個異常
- print (listA)
- print (listA.pop()) # pop 它會做兩件事：刪除 list 的最後一個元素，然後返回刪除元素的值。
- del listA[0] # 刪除列表中第一個元素
- print (listA)
- del listA[0:2] # 刪除列表中第一和第二個元素
- print (listA)
- # 迭代list
- for item in listA:
- 	print (item)

```
['a', 'b', 'c', 'd', 1, 2, 7, 8, ['a', 100]]
['a', 100]
['b', 'c', 'd', 1, 2, 7, 8]
['d', 1, 2, 7, 8]
d
1
2
7
8
```

List 的反轉序列與排序

- print ('原來list : ',listA)
- listA.reverse()
- print ('反轉list : ',listA)
- listA.pop()
- print (listA)
- #元素不相同則無法排序 'd' 和 數字
- listA.sort()
- print ('小到大排序 : ',listA)
- listA.sort(reverse=True)
- print ('大到小排序 : ',listA)
- print (max(listA))
- print (min(listA))
- print (sorted(listA))

```
原來list :  ['d', 1, 2, 7, 8]
反轉list :  [8, 7, 2, 1, 'd']
[8, 7, 2, 1]
小到大排序 :  [1, 2, 7, 8]
大到小排序 :  [8, 7, 2, 1]
8
1
[1, 2, 7, 8]
```

List 練習1

- 建立字串
  - A = list()
  - A = []
  - A = [1,2,3]
  - A = list(range(1,4))
  - A = list (“ABCD”)

```
In [1]: A=list()

In [2]: A
Out[2]: []

In [3]: A =[]

In [4]: A
Out[4]: []

In [5]: A = [1,2,3]

In [6]: A
Out[6]: [1, 2, 3]

In [7]: A = list(range(1,4))

In [8]: A
Out[8]: [1, 2, 3]
```

```
In [12]: A = list ("ABCD")

In [13]: A
Out[13]: ['A', 'B', 'C', 'D']
```

List 練習2

- String.format
- String.split ("sep")
  - "1 2 3".split()
  - "1,2,3".split(',')
- 內建函數
  - len(list ("ABCD"))
  - sum([1,2,3,4,5])
  - max([1,2,3,4,5])

```
In [16]: "1 2 3".split()
Out[16]: ['1', '2', '3']

In [17]: "1,2,3".split(',')
Out[17]: ['1', '2', '3']

In [18]: len(list ("ABCD"))
Out[18]: 4

In [19]: sum([1,2,3,4,5])
Out[19]: 15
```

```
In [29]: max([1,2,3,4,5])
Out[29]: 5
```

List 練習3

- import random
- a=[1,2,3,4,5]
- random.shuffle(a)
- a

```
In [25]: import random

In [26]: a=[1,2,3,4,5]

In [27]: random.shuffle(a)

In [28]: a
Out[28]: [5, 1, 4, 3, 2]
```

- 連接運算子 + , 重複運算子 *
  - [1,2,3]+['a', 'b', 'c', 'd']
  - 3*['a','b']

```
In [30]: [1,2,3]+['a', 'b', 'c', 'd']
Out[30]: [1, 2, 3, 'a', 'b', 'c', 'd']

In [31]: 3*['a','b']
Out[31]: ['a', 'b', 'a', 'b', 'a', 'b']
```

List 練習4

- 比較運算子
  - [1,2,3] > [1,2,4]
  - [1,2,3]!= [1,2,4]
  - [1,2,3]==[3,2,1]
  - [2,1,3]>[1,5,7]

```
In [34]: [1,2,3] > [1,2,4]
Out[34]: False

In [35]: [1,2,3]!= [1,2,4]
Out[35]: True

In [36]: [1,2,3]==[3,2,1]
Out[36]: False

In [37]: [2,1,3]>[1,5,7]
Out[37]: True
```

- in 與 not in運算子
  - 'a' in ['a','b','c']
  - 1 not in ['a','b','c']

```
In [38]: 'a' in ['a','b','c']
Out[38]: True

In [39]: 1 not in ['a','b','c']
Out[39]: True
```

作業4-2

- 請利用 IDLE 及 Spyder (擇一)完成數位化學習平台之作業
- 將結果上傳 .py 及程式結果截圖

下列 IPython 第一個執行 ex4-2sam.py

請修改 ex4-2sam.py 改成 ex4-2.py 結果如下圖

ex4-2.py 結果

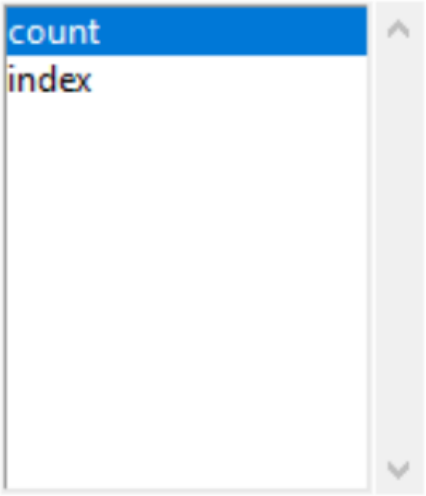
```
作業4-2 學號:xxx 姓名:xxx
B=[9,7,5,3,1,10,8,6,4,2]
max= 10
min= 1
小->大 B=[1,2,3,4,5,6,7,8,9,10]
大->小 B=[10,9,8,7,6,5,4,3,2,1]
```

序對(tuple)：使用小括號( )

- tuple使用( );
- 小括號裡面除了可以放各種不同型態的資料, 也可以放tuple;
- 資料的取出靠索引值,
- 資料儲存為有序的. 與 list 最大的不同在於 tuple 是唯獨且不可更變的資料結構.

Tuple 例子

```
>>> tuple1 = (20, 3.14, "三角形", 4+2j, [1,2,3], (4,5,6), 6+9j)
>>> tuple1
(20, 3.14, '三角形', (4+2j), [1, 2, 3], (4, 5, 6), (6+9j))
>>> tuple1.count(20)
1
>>> tuple1.
```



練習1

- 建立序對(tuple)
  - a=tuple((1,2,3))
  - a=(1,2,3)
  - a=tuple("ABCD")
  - a=tuple(range(1,6))
  - a=tuple(i*2 for i in range(1,4))

```
In [10]: a=tuple("ABCD")

In [11]: a
Out[11]: ('A', 'B', 'C', 'D')

In [12]: a=tuple(range(1,6))

In [13]: a
Out[13]: (1, 2, 3, 4, 5)

In [14]: a=tuple(i*2 for i in range(1,4))

In [15]: a
Out[15]: (2, 4, 6)
```

練習2:序對的運算(4-2tuple.py)

```
 8 a=(1,3,5)
 9 print (len(a))
10 print (max(a))
11 print (min(a))
12 print (sum(a))
13 print (a + tuple("ABCD"))
14 print (3*a)
15 print (a == (1,3,5))
16 print (a == (5,3,1))
17 print (a < (5,3,1))
18 a=tuple(range(1,10))
19 print (a)
20 print (a[2:5])
21 print (a[5:-1])
```

```
In [24]: runfile('D:/Users/yp719
PyNo4')
3
5
1
9
(1, 3, 5, 'A', 'B', 'C', 'D')
(1, 3, 5, 1, 3, 5, 1, 3, 5)
True
False
True
(1, 2, 3, 4, 5, 6, 7, 8, 9)
(3, 4, 5)
(6, 7, 8)
```

集合(set)：使用大括號{ }

- set使用{ };
- 大括號裡面不可以放的型態為 list 和 set
- 但可以放tuple;
- set的資料儲存為無序的且不重複
- 重複資料視為同一筆資料.

建立集合

- a=set()
- a=set({1,2,3})
- a={1,2,3}
- a=set("ABCDAB")
- a=set(range(1,6))
- a=set(i*2 for i in range(1,4))

```
set()
{1, 2, 3}
{1, 2, 3}
{'D', 'A', 'C', 'B'}
{1, 2, 3, 4, 5}
{2, 4, 6}
```

內建函式

- a={1,2,3,4,5}
- print(len(a))
- print(max(a))
- print(min(a))
- print(sum(a))

```
5
5
1
15
```

運算子

- 集合不支援 連接+ , 重複 * , 索引[ ] , 片段 [start:end]
- 集合支援 in , not in, ==, !=, > , >= ,< ,<=
- s1<=s2 表示 s1 為 s2 的子集合 (s2為s1的超集合)
- s1< s2 表示 s1 為 s2 的真子集合(s2為s1的真超集合)

集合處理方式

- 新增 set.add(x) 加入一個元素 x
- 取出 set.pop()
- 複製 set.copy()
- 刪除 set.clear()
- s1.issubset(s2) s1 為 s2 的子集合嗎?
- s1.isupperset(s2) s1 為 s2 的超集合嗎?

集合運算(4-3set.py)

- 交集 intersection
  - set.intersection(s)
  - set.intersection_update(s) 交集後更新set
- 聯集 union
  - set.union(s)
  - set.update(s) 聯集後更新set

```
21 s1={1,2,3}
22 s2={3,5,7}
23 s3=s1.union(s2)
24 print(s1) ;print(s2) ;print(s3)
25 s1.update(s2)
26 print(s1) ;print(s2) ;print(s3)
```

```
{1, 2, 3}
{3, 5, 7}
{1, 2, 3, 5, 7}
{1, 2, 3, 5, 7}
{3, 5, 7}
{1, 2, 3, 5, 7}
```

集合運算 (4-3set.py)

- 差集 difference ： set-s
  - set.difference(s)
  - set.difference_update(s)

```
13 s1={1,2,3,5}
14 s2={3,5,7,9}
15 s3=s1.difference(s2)
16 print(s1) ;print(s2) ;print(s3)
17 s3=s2.difference(s1)
18 print(s1) ;print(s2) ;print(s3)
19
20
```

```
{1, 2, 3, 5}
{9, 3, 5, 7}
{1, 2}
{1, 2, 3, 5}
{9, 3, 5, 7}
{9, 7}
```

集合的符號運算

Set Operation	Result
x in set	x 是否在 set 中
x not in s	x 是否不在 set 中
set1 & set2	輸出 set1 和 set2 的交集 (AND)
set1 | set2	輸出 set1 和 set2 的聯集 (OR)
set1 ^ set2	輸出 set1 和 set2 的對稱差集
set1 - set2	輸出 set1 和 set2 的差集
set1 < set2	判斷 set1 是否為 set2 的真子集 (proper subset)
set1 <= set2	判斷 set1 是否為 set2 的子集 (subset)
len(set)	回傳 set 的元素個數
min(set)	回傳 set 的最小值 (set中元素須為同型態)
max(set)	回傳 set 的最大值 (set中元素須為同型態)

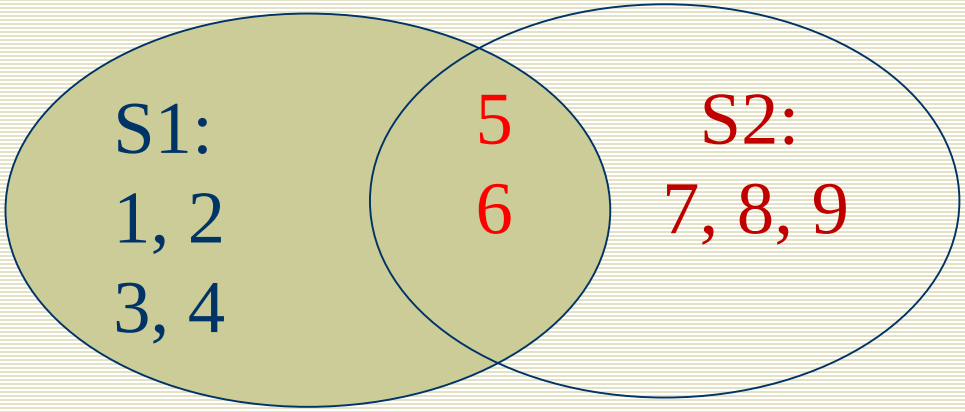
英文 vs 符號

Set method	description
set1.intersection(set2)	= set1 & set2
set1.union(set2)	= set1 | set2
set1.symmetric_difference(set2)	= set1 ^ sey2
set1.difference(set2)	= set1 - set2
set1.issubset(set2)	= set1 <= set2
set1.issuperset(set2)	= set1 >= set2
set1.isdisjoint(set2)	回傳 set1 和 set2 有無交集, 若無則為true
set.copy()	回傳 set 的複製集合
set.add(x)	增加 x 為 set 的元素
set.remove(x)	從 set 中移除 x 元素

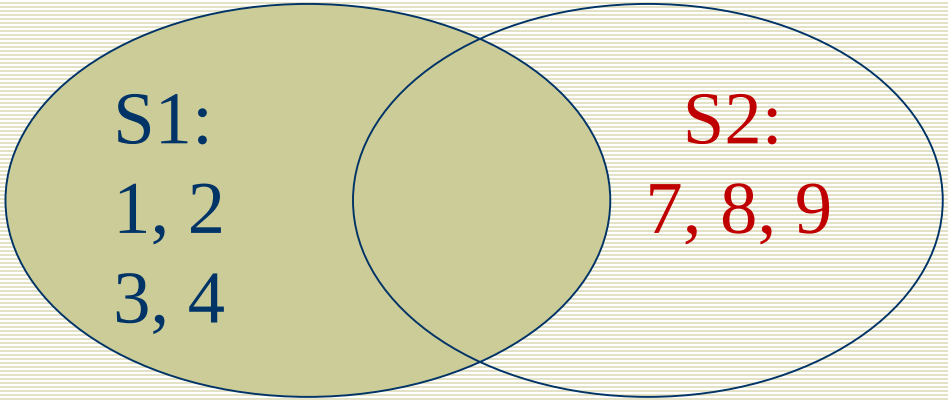
作業 4-3

- 請利用 IDLE 及 Spyder (擇一)完成數位化學習平台之作業
- 將結果上傳 .py 及程式結果截圖
- 提示: (S1∪S2) – (S1∩S2)

作業4-3



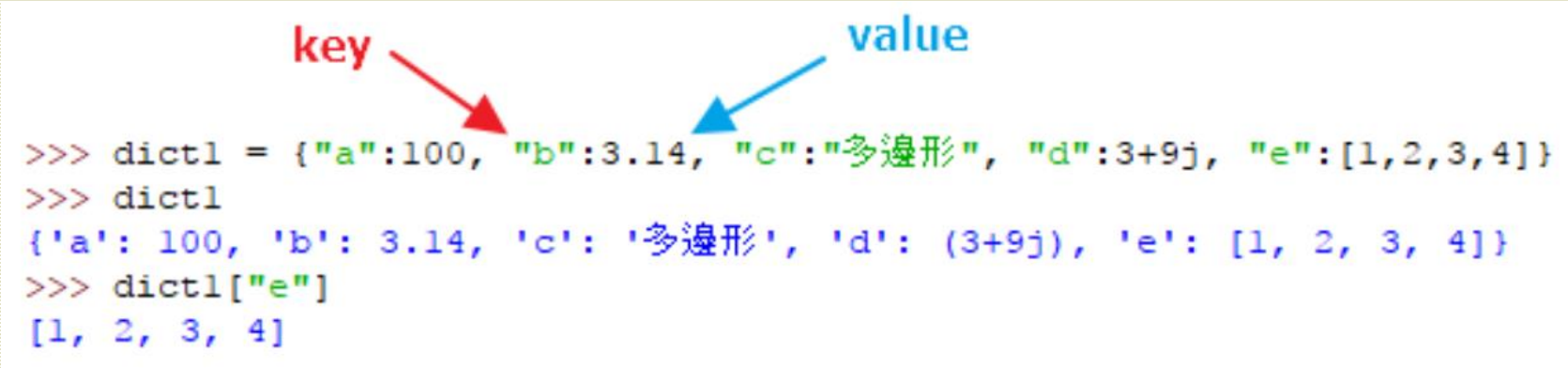
提示: $(S1 \cup S2) - (S1 \cap S2)$



字典(dict)：使用大括號{ }

- dict使用{ };
- 括號裡面可以放多種型態的資料;
- dict的型態為無序的, 使用key來檢索,
- key須為不可變的資料型態(ex:int, str)

dict 範例


key
value
>>> dict1 = {"a":100, "b":3.14, "c":"多邊形", "d":3+9j, "e":[1,2,3,4]}
>>> dict1
{'a': 100, 'b': 3.14, 'c': '多邊形', 'd': (3+9j), 'e': [1, 2, 3, 4]}
>>> dict1["e"]
[1, 2, 3, 4]

dict 運算

Dict Operation	Result
dict[key]	回傳 dict 中 key 的 value
dict[key]=value	指定 value 給 dict 中的 key
del dict[key]	刪除 dict 中 key 所對應的 value
key in dict	判斷 key 是否在 dict 中
key not in dict	判斷 key 是否不在 dict 中
iter(dict)	回傳 dict 的 key 建立的迭代器
len(dict)	回傳 dict 的配對資料個數

dict 函數

Dict method	description
dict.clear()	清空所有 dict 的資料
dict.copy()	複製 dict
dict.fromkeys(seq[, value])	以 seq 為 key, 賦予每個 key 相同 value
dict.get(key[, default])	回傳 dict 中 key 的值, 若無則回傳 default (=none)
dict.items()	回傳 dict_items 物件, 值為依序儲存的 key:value 序對.
dict.keys()	回傳 dict_keys 物件, 值為依序儲存的 key 序對.
dict.pop(key[, default])	移除 dict 中 key 的值, 若無則回傳 default
dict.popitem()	任意移除一組 key:value
dict.setdefault(key[, default])	若 key 在 dict 中, 則回傳 value; 若無則加入 key:default
dict1.update([dict2])	將 dict2 添加到 dict1
dict.values()	回傳 dict_values 物件, 值為依序儲存的 value 序對.

dict 例子

```
 8 d1 = {'id':5, 'name':'john', 'age':18, 8:'a number', (1, 2):100}
 9 d2 = dict(id=5, name='john', age=18)
10 print(d1)
11 print(d2)
12
13 print(d1['id'])
14 print(d1[8])
15 print(d1[(1, 2)])
16 print(d2['name'])
17 print(d2['age'])
```

```
{'id': 5, 'name': 'john', 'age': 18, 8: 'a number', (1, 2): 100}
{'id': 5, 'name': 'john', 'age': 18}
5
a number
100
john
18
```

```
{'id': 5, 'name': 'john', 'age': 18, 8: 'a number', (1, 2): 100, 'A': 65, 'C': 67, 'E': 69}
```

dict 例子 (4-4dict.py)

```
19 #d3 = {chr(c):c for c in range(65, 68)}
20 d3 = {chr(c):c for c in [65,67,69]}
21 print(d3)
22
23 for data in d3.keys():
24     print(data)
25 for i, data in enumerate(d3.keys()):
26     print(i, data)
27
28 for data in d3.values():
29     print(data)
30 for i, data in enumerate(d3.values()):
31     print(i, data)
32
33 for data in d3.items():
34     print(data)
35 for i, data in enumerate(d3.items()):
36     print(i, data)
37
38 z = {**d1, **d2, **d3}
39 print(z)
```

```
{'A': 65, 'C': 67, 'E': 69}
A
C
E
0 A
1 C
2 E
65
67
69
0 65
1 67
2 69
('A', 65)
('C', 67)
('E', 69)
0 ('A', 65)
1 ('C', 67)
2 ('E', 69)
```

```
{'id': 5, 'name': 'john', 'age': 18, 8: 'a number', (1, 2): 100, 'A': 65, 'C': 67, 'E': 69}
```