建立 list

在 Python 中，最常用來收集多個值的變數就是 list (串列)

語法：變數名稱 = [值1, 值2, 值3.....] (外面用中括號包住，值則以逗號分隔)

In [1]:

```
#建立list
colors = ['red','blue','yellow']

print(colors)
print(type(colors))
```

```
['red', 'blue', 'yellow']
<class 'list'>
```

在list 放入不同型別的值或子list

由於 list 可放入一群值，list也被稱為是個容器(container)，而其內部收集的值稱為元素(element)。元素不必是同一型別，甚至可以用另一個list當成元素

In [3]:

```
data = ['people', 3, [4.8, 'car', True]]
print(data)
```

```
['people', 3, [4.8, 'car', True]]
```

print()可直接印出整個list的內容。若想單獨取出list的特定元素，可以使用元素對應的索引(index)，搭配list變數名稱和中括號的方式來取出該元素：

語法：list [元素索引] 。ps：Python 容器索引一定從0開始，最後一個的索引會是[元素總數-1]

In [6]:

```
data = ['apple', 3, [4.5,'car',True]]

#用索引取出data的特定元素
print(data[0])
print(data[1])
print(data[2])
print()

#代表從data[2]子list中依索引取出元素，有兩個中括號索引時依續接起來即可
print(data[2][0])
print(data[2][2])
```

```
apple
3
[4.5, 'car', True]

4.5
True
```

Python中也可以用負數當索引，-1代表倒數第一個元素，-2是倒數第二個

In [9]:

```
print(data[-1])
print(data[-2])
print(data[-3])

#倒數第一個元素(子 list)內的倒數第一個元素
print(data[-1][-1])
```

```
[4.5, 'car', True]
3
apple
True
```

除了可以用索引取出個別元素，也可以用切片(slicing)從list中擷取出一部分元素，變成一個新的list：

語法：新 list = list [起點索引：終點索引 (不含)：間距] 。ps：如果這些參數省略，會直接套用預設值: 起點索引預設為 0, 終點索引預設為最末索引+1,間距則預設為 1

In [15]:

```
#list切片能讓我們從list容器擷取出特定範圍的元素。
c = ['a','b','c','d','e','f','g']

#整個list
print(c[:])
print()

#索引範圍 0-2
print(c[:3])
print()

#索引 3~最後
print(c[3:])
print()

#索引 2~4
print(c[2:5])
print()

#整個list，但每2個元素取一次
print(c[::2])
print()

#索引範圍 1~5，每2個元素取一次
print(c[1:6:2])
```

```
['a', 'b', 'c', 'd', 'e', 'f', 'g']

['a', 'b', 'c']

['d', 'e', 'f', 'g']

['c', 'd', 'e']

['a', 'c', 'e', 'g']

['b', 'd', 'f']
```

可以繼續在list內新增元素。兩種方法：

1. list.append(欲新增的值)。ps:新增到list最後，一次只能新增一個值

2. list += 新 list 。 ps: 一次能新增多個值(這些值得放在list內)

In [16]:

```
fruits = ['蘋果','香蕉','橘子']

#新增元素到最後頭
fruits.append('葡萄')
print(fruits)
print()

#繼續新增兩個元素到最後頭
fruits += ['鳳梨','草莓']
print(fruits)
```

```
['蘋果', '香蕉', '橘子', '葡萄']

['蘋果', '香蕉', '橘子', '葡萄', '鳳梨', '草莓']
```

list也能刪除元素。兩種語法：

1. list.remove(值)。ps:用remove()刪除指定的值

2. del list[索引]。ps:用del刪除指定的索引。也可以把當中的索引換成切片語法，就 能一次刪除多個值

In [17]:

```
fruits = ['蘋果','香蕉','橘子','葡萄','鳳梨','草莓']

#刪除索引 0 元素(蘋果)
del fruits[0]
print(fruits)
print()

fruits.remove('橘子')
print(fruits)

# 補充 : 使用remove()時，Python會從頭尋找元素，並將最先找到符合的值刪除。若後面有其他符合的元素，t
#        若想清空整個list的元素，可以呼叫list變數的clear()method
```

```
['香蕉', '橘子', '葡萄', '鳳梨', '草莓']

['香蕉', '葡萄', '鳳梨', '草莓']
```

建立dict (字典)

dict (dictionary,字典)和list一樣，能儲存多筆資料的容器。差異在於dict的每筆資料是以一組組鍵(key)和值(value)的形式構成:

語法：dict 變數名稱 = {鍵1:值1, 鍵2:值2, 鍵3:值3.....} 。ps: 1.用大括號包住 2.鍵與值用冒號配對 3. 各組資料以逗號分隔

由於dict的元素是鍵與值的組合，因此可以拿來當成對照表

In [18]:

```
capitals = {'瑞士':'伯恩','日本':'東京','美國':'華盛頓特區'}
print(capitals)
```

```
{'瑞士': '伯恩', '日本': '東京', '美國': '華盛頓特區'}
```

dict 的內容也可以由不同型別的資料構成:

In [19]:

```
#分行寫不影響內容，能讓鍵與值更容易閱讀
language = {'name':'Python',
            'version':'3.9',
            'types':['int','float','str','bool']}
print(language)

# ps: 能用其他dict當成某dict的值。但要注意的是，dict的鍵必須用內容固定的資料來建立，因此list和dict
```

```
{'name': 'Python', 'version': '3.9', 'types': ['int', 'float', 'str', 'boo
l']}
```

dict 取值時，要搭配中括號，但中括號內並不是填入索引，而是與該值配對的鍵：

語法：值 = dict[鍵] (注意: 若該鍵在dict中不存在，就會產生錯誤)

In [20]:

```
capitals = {'瑞士':'伯恩','日本':'東京','美國':'華盛頓特區'}

#傳入鍵
print(capitals['美國'])
print(capitals['日本'])
```

```
華盛頓特區
東京
```

In [22]:

```
capitals = {'瑞士':'伯恩','日本':'東京','美國':'華盛頓特區'}
print(capitals)
print()

#增加一組
capitals['越南'] = '河內'
print(capitals)
print()

#繼續增加兩組
capitals.update({'英國':'倫敦','法國':'巴黎'})
print(capitals)
```

```
{'瑞士': '伯恩', '日本': '東京', '美國': '華盛頓特區'}

{'瑞士': '伯恩', '日本': '東京', '美國': '華盛頓特區', '越南': '河內'}

{'瑞士': '伯恩', '日本': '東京', '美國': '華盛頓特區', '越南': '河內', '英國':
'倫敦', '法國': '巴黎'}
```

可以刪除dict 內的元素。兩種方法：

語法：1. del dict[鍵] 2. dict.pop(鍵)

In [24]:

```
capitals = {'瑞士':'伯恩','日本':'東京','美國':'華盛頓特區'}

#等同於呼叫 capitals.pop('美國')
del capitals['美國']
print(capitals)
print()

capitals.pop('瑞士')
print(capitals)
```

```
{'瑞士': '伯恩', '日本': '東京'}

{'日本': '東京'}
```

while 迴圈

重複執行某些程式碼，直到某個條件不再成立為止，可以使用while迴圈：

語法：

while 條件判斷式 : 程式區塊

ps : 1. while會先對條件式做判斷，如果條件為True，就執行接下來的程式區塊，然後再回到while做判斷，如此一直循環到條件式為False時，則結束迴 圈，然後繼續往下執行。 2. 在撰寫while迴圈時務必當心，因為若條件判斷式無論如何都為True的話，該迴圈就會變成無窮迴圈。這時就只能自行中止程式(按編輯器的停止鈕)

In [4]:

```
n = 0

#當n的值不超過5時就執行底下的程式
while n < 5:
    print(n)
    #n每次遞增1
    n += 1
```

```
0
1
2
3
4
```

使用break來脫離while迴圈

除了用條件判斷式來控制迴圈重複執行與否，另一個辦法是在迴圈內使用break敘述來打斷迴圈 (通常是搭配if來檢查某個條件是否成立):

語法：

while 條件判斷式: 程式區塊 if 條件判斷式: break

In [5]:

```
n = 0

#while迴圈本身為無窮迴圈
while True:
    print(n)
    n += 1
    #在n >=5 時中止 while迴圈
    if n >=5:
        break
```

```
0
1
2
3
4
```

希望在這一輪迴圈跳過某個位置之後的程式碼時，可以使用continue敘述：

語法：

while 條件判斷式: 程式區塊 if 條件判斷式:
continue

要跳過的程式區塊

In [8]:

```
n = 1
while n < 10:

    #每次n遞增1
    n += 1
    if n %3 ==0:

        #若n是3的倍數就跳過下面的程式碼(不印)
        continue
    #印出n的值
    print(n)

# ps : 當n是3的倍數時，後面的print(n)就會被跳過、不會印出n的值，然後回到while的條件判斷式繼續執行
```

```
2
4
5
7
8
10
```

基本for迴圈

迴圈除了用來重複特定的程式碼，也有一個很常用的用途是拿來逐次處理容器內的元素，此種情形下，改用for迴圈會更為合適

語法：

for 變數 in 容器 : 程式區塊

ps : for 迴圈每次重覆時，會從容器取出一個元素的值、指派給for後面變數。等到容器已經無值可取時，迴圈就會停止。逐一拜訪所有元素的動作便稱為走訪(iterate)或迭代

In [9]:

```
animal_list = ['dog','cat','monkey','bird','elephant']

#每次從animal_list取出一個值指派給變數animal
for animal in animal_list:
    print(animal)
```

```
dog
cat
monkey
bird
elephant
```

走訪二維list

若要用for迴圈走訪二維list (一個list的每個元素都是子list)，可以用內外兩層或稱巢狀(nested)的for迴圈來走訪

In [10]:

```
fruits = [
    ['apple','red'],
    ['banana','yellow'],
    ['guava','green'],
]

#第一層for迴圈(走訪 list)
for fruit in fruits:
    print(fruit)
```

```
['apple', 'red']
['banana', 'yellow']
['guava', 'green']
```

In [12]:

```
#題目說明 :下面fruits容器中，每個元素又各是有兩個元素的list。用第一層for迴圈走訪fruits時，得到的就
#          用第二層for迴圈來走訪子元素。好逐次取出子元素list中的元素

fruits = [
    ['apple','red'],
    ['banana','yellow'],
    ['guava','green'],
]

for fruit in fruits:
    #第二層for迴圈(走訪子list)
    for item in fruit:
        print(item)
```

```
apple
red
banana
yellow
guava
green
```

for 迴圈進階用法

用for走訪list的一個缺點是，儘管它會自動走訪完所有元素，卻無法得知該元素的索引。但有時，索引也是很有用處的資訊，例如能代表元素的順序等等

解決方式之一是用range()函式產生數列，長度跟我們想走訪的目標list一樣

語法：

for 索引 in range(N): 程式區塊

ps : N 為list的長度，有5個元素其長度就是5。range(N)會傳回0至N-1的數列

In [13]:

```
#由於range()傳回的不是一般容器，要先轉換成list型別才看得到元素
print(list(range(5)))
```

```
[0, 1, 2, 3, 4]
```

In [2]:

```
animal_list = ['dog','cat','monkey','bird','elephant']
for index in range(5):
    print('item', index,'=',animal_list[index])

#ps:第一個index是逐一走訪索引 0、1、2、3、4。第二個index是代入
```

```
item 0 = dog
item 1 = cat
item 2 = monkey
item 3 = bird
item 4 = elephant
```

在for 迴圈搭配enumerate()來同時走訪索引及元素

前面為了取得容器索引，我們要使用range()來額外產生一個數列，而且還要確保數列和你的容器長度一致，其實這是其他程式語言的習慣，Python有更好的做法

可以改用enumerate(),它能將list容器的值包裝成(索引，值)的形式傳回，使得for迴圈走訪時不必設定要走訪的長度，也能取得每個元素的索引

語法：

for index, item in enumerate(list 容器): 程式區塊

In [3]:

```
# ps: 同樣的，這裡先將enumerate()的結果轉成list來觀看內容
print(list(enumerate(animal_list)))

#答案所產生的結果，每個元素現在變成有2個項目的子容器，包括索引和值，索引序號是enumerate()產生的
#上面用小括號而不是中括號括起來的容器叫做tuple，等於是元素不可改變的list
```

```
[(0, 'dog'), (1, 'cat'), (2, 'monkey'), (3, 'bird'), (4, 'elephant')]
```

In [4]:

```
animal_list = ['dog','cat','monkey','bird','elephant']

#每次將enumerate()傳回的子容器的內容分別指派給index和item
for index, item in enumerate(animal_list):
    print('item',index,'=',item)
```

```
item 0 = dog
item 1 = cat
item 2 = monkey
item 3 = bird
item 4 = elephant
```

用zip()同時走訪多個list

如手上有多個長度相同的list，每次想各從一個list中取出一個元素來，可以使用zip()函式搭配for迴圈:

語法：

for item1, item2 in zip(list 容器1, list容器2): 程式區塊

#ps : zip()會每次讀取這些容器中的1個元素，並打包成zip()容器傳回

In [5]:

```
index_list = ['a','b','c','d','e']
animal_list = ['dog','cat','monkey','bird','elephant']

print(list(zip(index_list, animal_list)))

#zip()處理完是個zip()容器，將其轉換為list
#從答案可以看出兩個list的元素被拆解，變成兩兩成對放在一起
```

```
[('a', 'dog'), ('b', 'cat'), ('c', 'monkey'), ('d', 'bird'), ('e', 'elephan
t')]
```

In [6]:

```
fruits = ['apple','peach','banana','guava','papaya']
colors = ['red','pink','yellow','green','orange']

for name, color in zip(fruits, colors):
    print(name, 'is',color)
```

```
apple is red
peach is pink
banana is yellow
guava is green
papaya is orange
```

用 for 走訪 dict

前面我們討論的走訪對象都是list，不過用for迴圈走訪dict容器時就比較特殊了，必須使用dict物件.item()來取得每一組資料的鍵與值：

語法：

for key , value in dict物件item(): 程式區塊

ps :如果不使用items()而直接走訪dict的話，只會得到dict的所有鍵而已

In [2]:

```
fruits = {
    'apple': 'red',
    'peach': 'pink',
    'banana': 'yellow',
    'guava': 'green',
    'papaya': 'orange'
}

#只使用dict，沒有使用items()
print(list(fruits))
```

```
['apple', 'peach', 'banana', 'guava', 'papaya']
```

In [3]:

```
fruits = {
    'apple':'red',
    'peach':'pink',
    'banana':'yellow',
    'guava':'green',
    'papaya':'orange'
}

print(list(fruits.items()))
```

```
[('apple', 'red'), ('peach', 'pink'), ('banana', 'yellow'), ('guava', 'gree
n'), ('papaya', 'orange')]
```

In [4]:

```
fruits = {
    'apple': 'red',
    'peach': 'pink',
    'banana': 'yellow',
    'guava': 'green',
    'papaya': 'orange'
}

#用for迴圈走訪dict的所有鍵與值
#逐一走訪dict將鍵存入name，將值存入color
for name, color in fruits.items():
    print(name,'is',color)
```

```
apple is red
peach is pink
banana is yellow
guava is green
papaya is orange
```