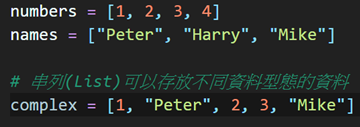
Python List

Python是個有大小寫敏感性(Case-sensitive)的語言，這表示在程式裡面使用相同字母但不同的大小寫組合成的變數是不相同的。

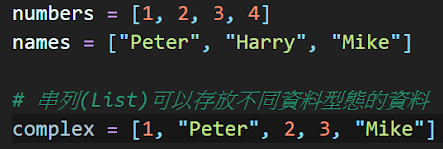
List(串列)是一個Python非常重要的資料型態，它就像是一個容器，可以用來存放多個不同資料型態的資料(元素)，以逗號分隔並且用 [] 符號將所有元素括起



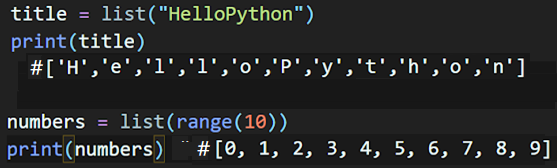
* 串列有幾個特性：

1. Iterable(可疊代的)：所以上一篇文章介紹的Python迴圈可以應用在串列上。
2. Modifiable(可修改的)：串列中的元素可以透過Python提供的串列方法(Method)來進行修改。
3. **建立串列的方法**

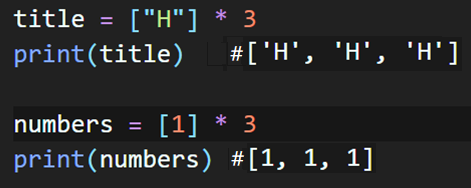
1.    直接於 [] 符號中輸入元素資料。



2.   使用Python的list()方法，傳入Iterable(可疊代的)物件來建立串列。

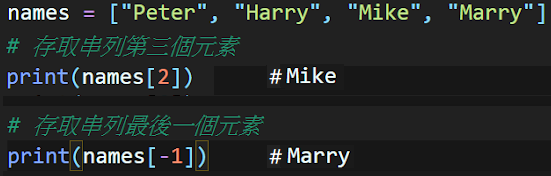


3.    使用 \* 符號來建立多個相同元素的串列



1. **存取串列元素的方法**

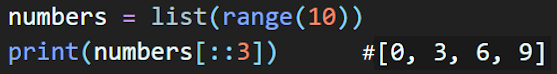
1.    Python存取串列元素的方式和[**字串**](https://learncodewithmike.blogspot.com/2019/11/python4-python.html)的操作一樣，使用 [] 符號並傳入索引值(同樣從0開始計算)。



2.    如果想取得特定範圍的串列元素，和[**字串**](https://learncodewithmike.blogspot.com/2019/11/python4-python.html)一樣使用 [:] 符號並傳入索引值。

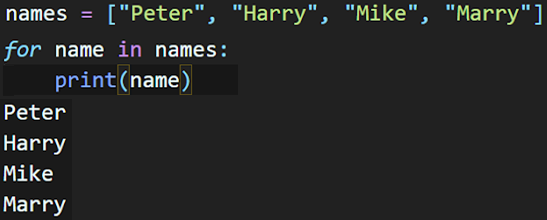


1. Python串列還有一個特別的存取方式，就是使用 [::遞增(減)值]，範例如下：



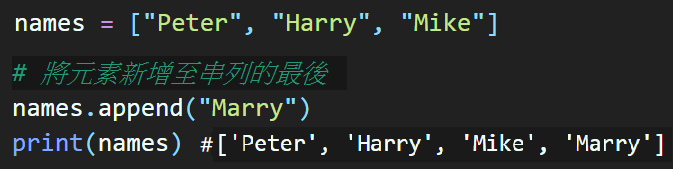
從範例中可以看到，串列的索引值從0開始，依據遞增值3，取出每隔3的索引值元素。

4.    除了使用 [] 符號存取串列元素外，也可以透過[**Python迴圈**](https://learncodewithmike.blogspot.com/2019/12/python.html)來讀取串列中的每一個元素，因為串列也是Iterable(可疊代的)，如下範例：

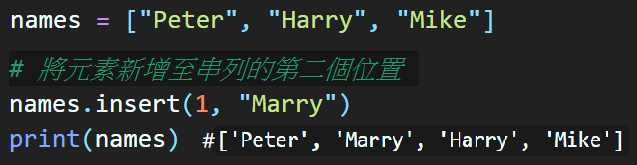


1. **新增串列元素的方法 (append() 與 extend())**

1.    使用append()方法，將元素新增至串列的最後。



2.    使用insert()方法，將元素增至串列的特定位置，注意Python串列位置索引值從0開始。



**extend()與append()** 的區別

list1=[1,2,3,4,5,6]

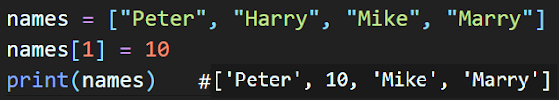
list2=[1,2,3,4,5,6]

list1.append([7,8]) #[1, 2, 3, 4, 5, 6, [7, 8]]

list2.extend([7,8]) #[1, 2, 3, 4, 5, 6, 7, 8]

1. **修改串列元素的方法 (**直接指派修改)

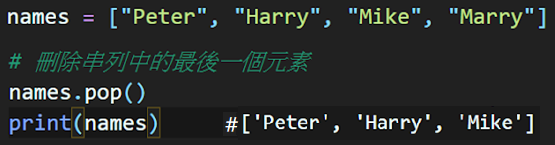
首先使用 [] 符號存取想修改的索引值，接著指派新的值，範例如下：

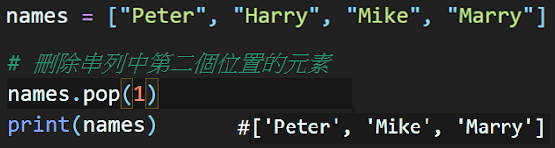


範例中，將串列的第二個元素修改為整數10。#可以直接指派修改

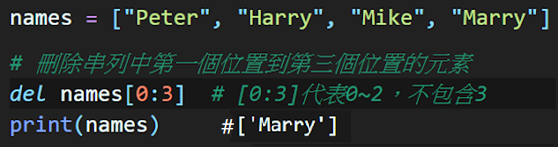
1. **刪除串列元素的方法** pop()、pop(x)、remove(obj)、del()

1.  使用pop()方法，將串列的最後一個元素刪除。如果想刪除特定位置的元素，則傳入位置索引值。

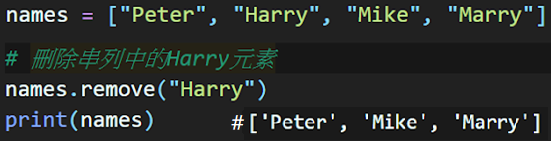




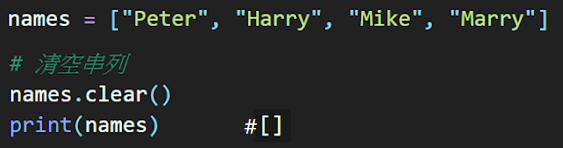
2.   刪除特定範圍的元素，可以使用**del ()**指令，指定要刪除的範圍位置索引值。



3.  當不知道元素的位置索引值，可以使用**Remove()**方法(方法內需有參數)，傳入想刪除的元素。注意如果此元素在串列中有多個(Harry)，Remove()方法只會刪除第一個出現的。

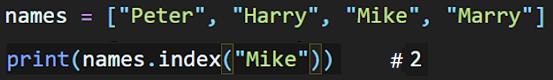


4.    清空串列可以使用clear()方法。

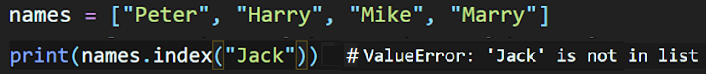


1. **尋找串列元素的方法**

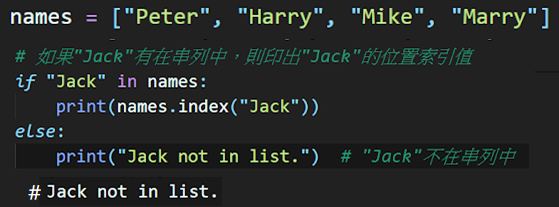
1.    **使用index()方法**，可以將要尋找的串列元素傳入，它會回傳該元素的位置索引值，如下範例：



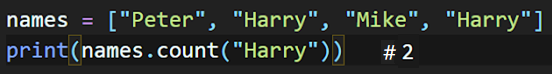
如果要尋找的串列元素不在串列中，則會出現錯誤訊息(不會傳回-1)，如下範例：



所以比較好的寫法是在取得元素的索引值前，先判斷該元素是否在串列中，如下範例：



2.    使用count()方法，可以將要尋找的串列元素傳入，它會回傳該元素在串列中的個數。



From：<https://www.learncodewithmike.com/2019/12/python-list.html>

1. **串列排序的方法**

list1=[1, 3, 2, 4, 6, 5]

list1.reverse()

print(list1) # [5, 6, 4, 2, 3, 1]

list1=[1, 3, 2, 4, 6, 5]

list1.sort()

print(list1) #[1, 2, 3, 4, 5, 6]

1. **串列切片**

**List Slicing 串列切片**

list = [1,2,3,4,5,6]

**list[:]** #全部

list2 = list[:] #始、末都不指定，複製全部[1, 2, 3, 4, 5, 6]

**list[::x]** #第一項開始, 間格 x 到最末

list2 = list[::2] #第一項開始，間隔一個 [1, 3, 5]

**list[::-x]** #最後一項開始，間隔 x 到最開始

list2 = list[::-2] #最後一項開始，間隔一個 [6, 4, 2]

list2 = list[::-1] #最後一項開始，逐項取得 [6, 5, 4, 3, 2, 1]

#slicing from the beginning to index stop – 1

**list[x:]** # x(index) 前的項數；

list2 = list[:2] #前兩項資料 [1, 2] list[:x] -> index 0,1

list2 = list[:3] #前三項資料[1, 2, 3] -> index 0,2

**list[-x:]** #後x的項數；

list2 = list[-2:] #後兩項資料 [5, 6] -> index[4,5]

list2 = list[-3:] #後三項資料 [4, 5, 6] -> index[3,4,5]

# slicing from index start to the end

**list[star:]**  #star開始(index) 取到 最末；

list2= list[2:] # [3, 4, 5, 6] list[start:] start -> end

list2= list[3:] # [4, 5, 6] list[start:] start -> end

https://ithelp.ithome.com.tw/articles/10204734

**list[star:-stop]**  #star開始 到 末stop的index；

list2 = list[3:-2] #按照索引值取 [4] 4開始到末項-2，index[3]

list2 = list[2:-2] #按照索引值取 [3, 4] 3開始到末項-2，index[2,3]

list2 = list[2:-1] #[3, 4, 5] 3開始到末項-1，index[2,3,4]

list2 = list[2:-3] #[3] 3開始到末項-3，index[2]

list2 = list[2:-4] #[] 3開始到末項-4，[]

list2 = list[1:-2] #[2, 3, 4] 2開始到末項-2，index[1,2,3]

**list[star:stop]**  #範圍內的index；

list2 =list[0:1] #[1] 可逐一取

list2=list[1:2]) #[2] 可逐一取

list2=list[1:3]) #[2] 可取多項

1. list 與 tuple 互轉

list1=[1,2,3,4,5,6]

tuple1=tuple(list1) #(1, 2, 3, 4, 5, 6)

tuple2=(1,2,3,4,5,6,7,8)

list12=list(tuple2) #[1, 2, 3, 4, 5, 6,7,8]

**常用方法**

| **方法** | **描述** | **範例 fruits = [‘apple’, ‘banana’];** |
| --- | --- | --- |
| append() | 在列表末尾添加元素 | fruits.append(‘orange’) → [‘apple’, ‘banana’, ‘orange’] |
| extend() | 將可迭代的元素添加到列表末尾 | fruits.extend([‘orange’, ‘grape’]) → [‘apple’, ‘banana’, ‘orange’, ‘grape’] |
| insert() | 在指定位置插入元素 | fruits.insert(1, ‘orange’) → [‘apple’, ‘orange’, ‘banana’] |
| remove() | 移除第一次出現的指定元素 | fruits.remove(‘banana’) → [‘apple’] |
| pop() | 移除指定位置的元素（預設為最後一個元素），並返回該元素 | removed\_fruit = fruits.pop(1) → removed\_fruit = ‘banana’; fruits = [‘apple’] |
| clear() | 清空列表中的所有元素 | fruits.clear() → [] |
| index() | 返回第一次出現指定元素的索引，如果不存在則引發 ValueError | index = fruits.index(‘banana’) → index = 1 |
| count() | 返回指定元素在列表中出現的次數 | count = fruits.count(‘banana’) → count = 2 |
| sort() | 對列表進行排序（預設遞增排序），原地排序 | fruits.sort() → [‘apple’, ‘banana’] |
| reverse() | 將列表中的元素反向排序，原地操作 | fruits.reverse() → [‘banana’, ‘apple’] |
| copy() | 複製列表 | new\_fruits = fruits.copy() → new\_fruits = [‘apple’, ‘banana’] |

列表是一種非常有用的資料類型，它可以用來儲存各種不同類型的元素。你可以使用這些操作來創建、修改、添加、刪除和查詢列表中的元素。

**Tuple 元組**

在Python中，元組（Tuple）是另一種常用的資料類型，用於儲存一組有序的元素。不同於列表，元組是不可變的（Immutable），這意味著一旦創建了元組，就無法修改其內容。元組使用圓括號 () 來表示，元素之間用逗號 , 分隔。以下是元組常用的操作：

**創建**

fruits\_tuple = ('apple', 'banana', 'orange', 'grape')

numbers\_tuple = (1, 2, 3, 4, 5)

mixed\_tuple = (1, 'apple', True, 3.14)

**獲取**

fruits\_tuple = ('apple', 'banana', 'orange', 'grape')

print(fruits\_tuple[0]) # 輸出：'apple'

print(fruits\_tuple[2]) # 輸出：'orange'

**不可改變**

fruits\_tuple = ('apple', 'banana', 'orange', 'grape')

# 以下操作會引發 TypeError: 'tuple' object does not support item assignment

fruits\_tuple[1] = 'pear'

del fruits\_tuple[2]

**元組解包**

fruits\_tuple = ('apple', 'banana', 'orange', 'grape')

fruit1, fruit2, fruit3, fruit4 = fruits\_tuple

print(fruit1) # 輸出：'apple'

print(fruit2) # 輸出：'banana'

**元組長度**

fruits\_tuple = ('apple', 'banana', 'orange', 'grape')

length = len(fruits\_tuple)

print(length) # 輸出：4

**元組檢查**

fruits\_tuple = ('apple', 'banana', 'orange', 'grape')

print('banana' in fruits\_tuple) # 輸出：True

print('watermelon' not in fruits\_tuple) # 輸出：True

**常用方法**

| **方法** | **描述** | **範例** |
| --- | --- | --- |
| count() | 返回指定元素在元組中出現的次數 | fruits = (‘apple’, ‘banana’, ‘banana’, ‘orange’); count = fruits.count(‘banana’) → count = 2 |
| index() | 返回第一次出現指定元素的索引，如果不存在則引發 ValueError | fruits = (‘apple’, ‘banana’, ‘orange’); index = fruits.index(‘banana’) → index = 1 |
| len() | 獲取元組的長度 | fruits = (‘apple’, ‘banana’, ‘orange’); length = len(fruits) → length = 3 |
| sorted() | 返回排序後的元組（由於元組是不可變的，這實際上創建了一個新元組） | fruits = (‘orange’, ‘banana’, ‘apple’); sorted\_fruits = sorted(fruits) → sorted\_fruits = (‘apple’, ‘banana’, ‘orange’) |
| reversed() | 返回反向的元組（同樣是創建了一個新元組） | fruits = (‘apple’, ‘banana’, ‘orange’); reversed\_fruits = tuple(reversed(fruits)) → reversed\_fruits = (‘orange’, ‘banana’, ‘apple’) |
| tuple() | 將可迭代的元素轉換為元組 | fruits\_list = [‘apple’, ‘banana’, ‘orange’]; fruits\_tuple = tuple(fruits\_list) → fruits\_tuple = (‘apple’, ‘banana’, ‘orange’) |

元組的不可變性使其在某些情況下比列表更適合使用，特別是在希望保證資料的不可變性時。你可以使用這些操作來創建、查詢元組中的元素以及進行元組解包。

**List 和 Tuple 比較**

列表（List）和元組（Tuple）是Python中兩種常用的資料結構，它們都用於儲存一組有序的元素。然而，它們之間有幾個主要的不同點：

1. **可變性：**

列表是可變的資料類型，這意味著你可以在創建後修改列表的內容，添加、刪除或修改元素。

元組是不可變的資料類型，一旦創建後就不能修改的內容，元組中的元素是固定的。

1. **創建方式：**

列表使用方括號 [] 來創建，元素之間用逗號 , 分隔。

元組使用圓括號 () 來創建，元素之間用逗號 , 分隔。

1. **性能：**

元組比列表更輕量，因為元組是不可變的，所以它們需要更少的內存空間和處理時間。

列表由於是可變的，可能需要更多的內存空間和處理時間。

1. **適用情況：**

使用列表當你需要在資料結構中添加、刪除或修改元素時，或者需要保持順序且元素可能重複出現。

使用元組當你需要保護資料免於被意外修改，或者需要在多個函式之間傳遞不可變的資料結構。

##遍歷列表元素

[1, 3, 300, 7, 100]

#透過迴圈用下標遍歷列表元素

for index in range(len(list1)):

print(list1[index])

1

3

300

7

100

#透過for迴圈遍歷列表元素

for elem in list1:

print(elem)

1

3

300

7

100

#透過enumerate函式處理列表之後再遍歷可以同時獲得元素索引和值

for index, elem in enumerate(list1):

print(index, elem)

0 1

1 3

2 300

3 7

4 100

#動態建立list 或 tuple

a=list(range(0,10)) #[0 1 2 3 4 5 6 7 8 9]

print (3 in a) #判斷是否在其中

b=tuple(range(0,10))

print(5 in b, b)