Python 程式設計：元組

目錄

1. 基本概念

元組（tuple）是一種用於儲存多個項目（元素）的數據結構。以是不同類型，例如數字、字串、甚至是另一個元組。元組與列表（List）最主要的區別在於：元組是不可變的 (immutable)，也就是說，一旦建立，它的內容就不能被修改、增加或刪除。

```
my_tuple = (1, 2, 'a', 'b')
# 只有一個元素的元組： 必須在元素後加上一個逗號，否則它會被視為普通變數。
# 正確： my_tuple = (1,)
# 錯誤： my_tuple = (1)
```

```
# 定義一個元組
numbers = (1, 2, 3, 4, 5)

# 定義一個包含不同類型元素的元組
fruits = ("apple", "orange")

# 使用 type() 函數檢查變數類型
print(type(numbers))   # 輸出: <class 'tuple'>
print(type(fruits))    # 輸出: <class 'tuple'>
```

2. 元組的功能 (Functionality of a Tuple)

- 更安全的保護資料：當你確定某些資料永遠不應被更改時（例如，圖像的長寬、某個像素的顏色值），將其儲存在元組中可以防止意外修改，使程式碼更穩定、更安全。
- 元組比列表更輕量、簡單，佔用的系統資源較少，因此在某些情況下，元組的執行速度會比列表快。
- 儲存不可變的數據集。例如圓周率 pi、自然常數 e、黃金比例等科學常數。

```
constants = (3.14159, 2.71828, 1.61803)
```

- 儲存大量座標數據：座標數據通常是不可變的，每個點可以儲存在一個元組中，然後將多個元組儲存在一個列表中。

```
coordinates_list = [(x, y) for x in range(100) for y in range(100)]
```

- 函數返回多個值：函數可以只用一個 return 語句返回多個值，此時 Python 會自動將這些值打包成一個元組。

```
def get_min_max(numbers):
  return min(numbers), max(numbers)

min_value, max_value = get_min_max([1, 5, 9, 3, 7, 6])
print(f'Min: {min_value}, Max: {max_value}') # 輸出: Min: 1, Max: 9
```

3. 讀取元組(與 list 相同)

元組的索引值從 0 開始。

```
fruits = ('apple', 'orange')
print(fruits[0])  # 讀取第一個元素
print(fruits[1])  # 讀取第二個元素
```

4. 遍歷所有元組元素(與 list 相同)

你可以使用 for 迴圈來遍歷元組中的所有元素。

```
keys = ('magic', 'xaab', 9099)
for key in keys:
  print(key)
```

5. 元組切片 (Tuple Slices) (與 list 相同)

切片的語法與列表相似：my_tuple[start:end]

6. 修改元組內容會產生錯誤

這是 tuple 與 list 最大的不同點，tuple 的內容是不可變的。嘗試修改、增加或刪除元組的元素都會導致錯誤。

```
fruits = ('apple', 'orange')
```

```
# 嘗試修改元組的元素
fruits[0] = 'watermelon'
# TypeError: 'tuple' object does not support item assignment
```

雖然元組本身不能修改，但如果你確實需要修改內容，你可以變通地創建一個新的元組。

方法一： 將 tuple 轉換成 list，修改列表，再將列表轉換回元組。

```
original_tuple = (10, 20, 30)
temp_list = list(original_tuple) # 轉換成 list
temp_list[1] = 99 # 修改 list 中的元素
modified_tuple = tuple(temp_list) # 再轉回 tuple
```

方法二： 創建一個新的 tuple，將舊 tuple 的元素和新元素合併。

```
original_tuple = (1, 2, 3)
new_element = 4 # 新元素
new_tuple = original_tuple + (new_element,) # 合併成新的元組
print(new_tuple)
```

7. enumerate 使用於 tuple

enumerate 會產生一對一對的（索引, 元素）組合，方便你在迴圈中同時獲得元素的索引和值。

```
enumerate(可迭代物件, start=0)
```

7.1 範例：

```
drinks = ('coffee', 'tea', 'wine')
enumerate_drinks = enumerate(drinks) # 使用 enumerate() 函數
print(enumerate_drinks)  # 打印 enumerate 物件，會顯示其記憶體位置

# 將 enumerate 物件轉為列表，方便查看內容
print(list(enumerate_drinks)) # 輸出: [(0, 'coffee'), (1, 'tea'), (2, 'wine')]
```

7.2 範例：

```
# 在 for 迴圈中使用 enumerate
drinks = ('coffee', 'tea', 'wine')

# 在 for 迴圈中同時獲取索引 (i) 和元素 (drink)
for i, drink in enumerate(drinks):
  print(f"{i}: {drink}")
```

8. zip() 函數打包多個物件

zip() 是一個內建函數，它可以將多個可迭代物件（長度不一定要相同）中的對應元素打包成一個元組序列。如果物件長度不一致，zip() 會以最短的那個為準。

```
zip(可迭代物件 1, 可迭代物件 2, ...)
```

8.1 範例：

```
fields = ('Name', 'Age', 'Hometown')
info = ('Peter', '30', 'Chicago')
zipdata = zip(fields, info) # 使用 zip() 打包兩個元組

# 打印 zip 物件
print(zipdata) # 輸出: <zip object at 0x...>

# 將 zip 物件轉換為列表，方便查看
player = list(zipdata)
print(player) # 輸出: [('Name', 'Peter'), ('Age', '30'), ('Hometown', 'Chicago')]
```

8.2 範例：長度不一致的例子

```
fields = ('Name', 'Age', 'Hometown')
info = ('Peter', '30') # 只有兩個元素

zipdata = zip(fields, info)
player = list(zipdata)
print(player) # 輸出: [('Name', 'Peter'), ('Age', '30')]，'Hometown' 被忽略了，因為 info 元組沒有對應的元素。
```