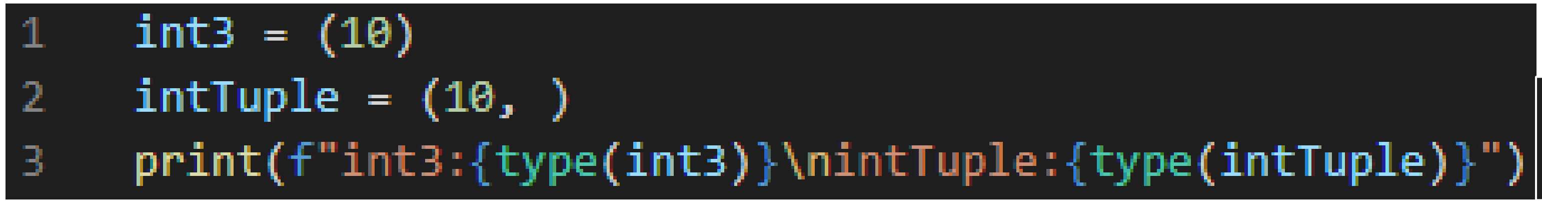
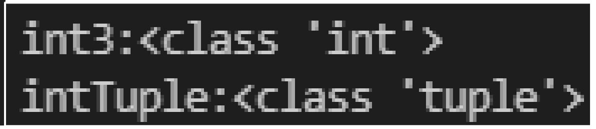
Chapter 8
元組 (Tuple)

8.1 元組的定義

- Python提供一種資料型態稱元組(tuple) 。
  - 這種資料型態結構與串列完全相同。
  - 與串列最大的差異是，它的元素值與元素個數不可更動，有時又可稱不可改變的串列。
  - 串列將元素放在 [] 中，元組則使用小括號 "()" 。
    - myTuple = (元素1, 元素2, … , 元素n)
    - 如果元組內的元素只有一個，在定義時需在元素右邊加上逗號。

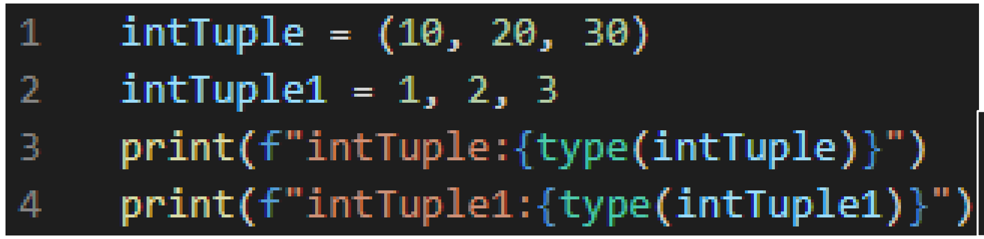


```python
int3 = (10)
intTuple = (10, )
print(f"int3:{type(int3)}\nintTuple:{type(intTuple)}")
```

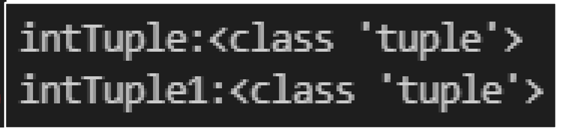


```
int3:<class 'int'>
intTuple:<class 'tuple'>
```

    - 也可將小括號拿掉



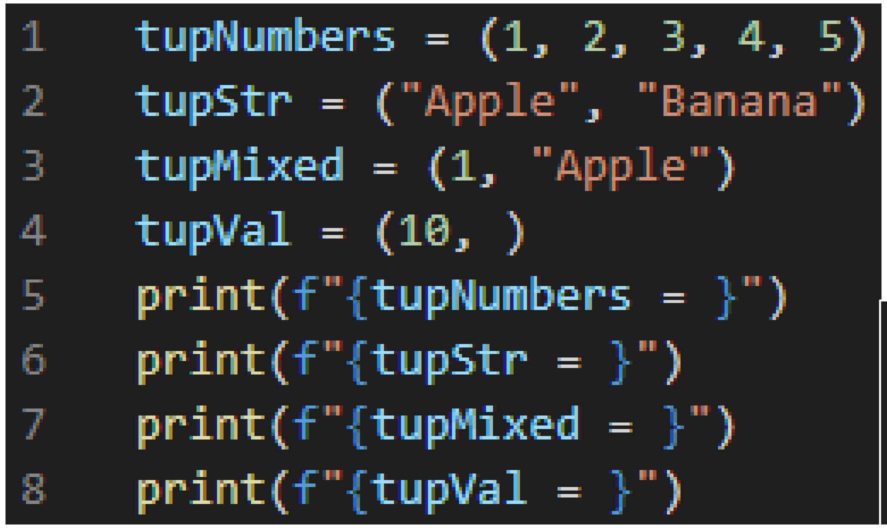
```python
intTuple = (10, 20, 30)
intTuple1 = 1, 2, 3
print(f"intTuple:{type(intTuple)}")
print(f"intTuple1:{type(intTuple1)}")
```



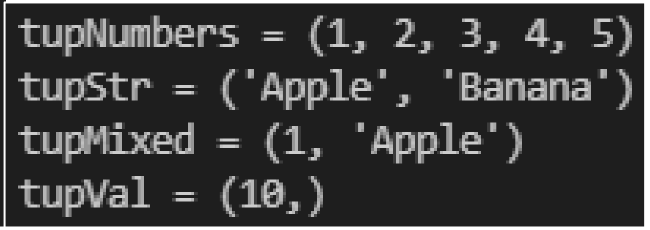
```
intTuple:<class 'tuple'>
intTuple1:<class 'tuple'>
```

8.1 元組的定義

- Python提供一種資料型態稱元組(tuple) 。



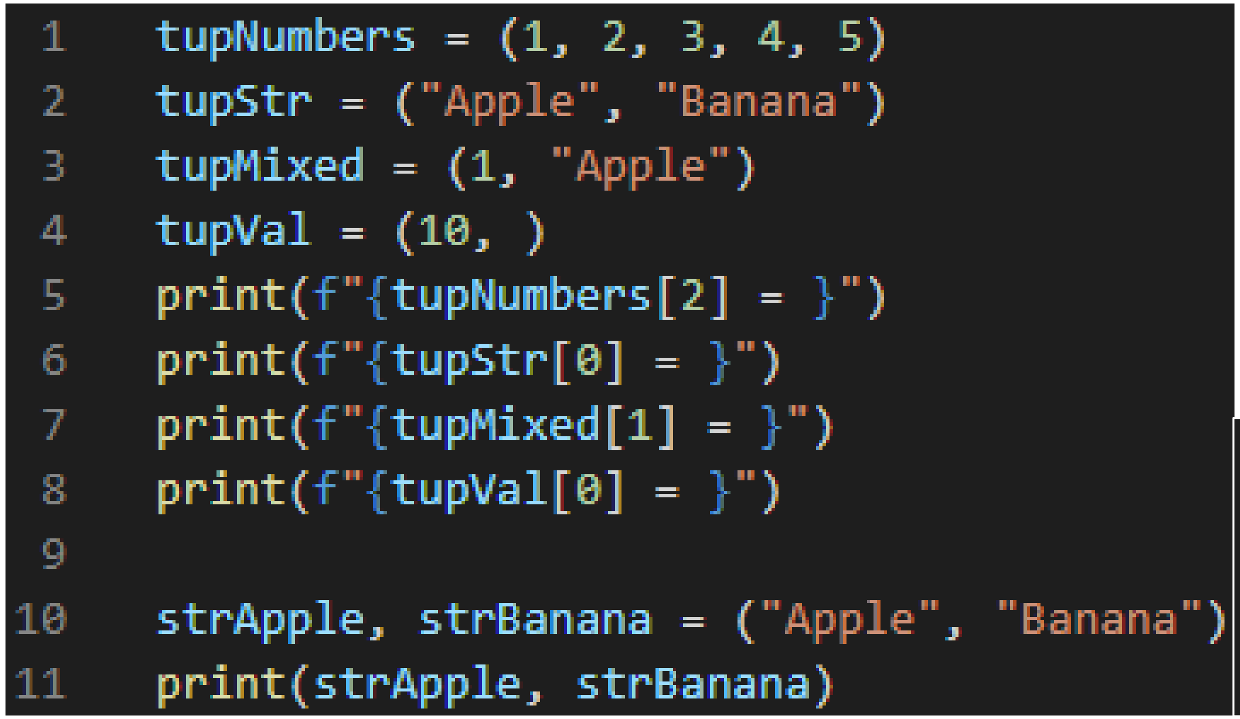
```python
tupNumbers = (1, 2, 3, 4, 5)
tupStr = ("Apple", "Banana")
tupMixed = (1, "Apple")
tupVal = (10, )
print(f"{tupNumbers = }")
print(f"{tupStr = }")
print(f"{tupMixed = }")
print(f"{tupVal = }")
```



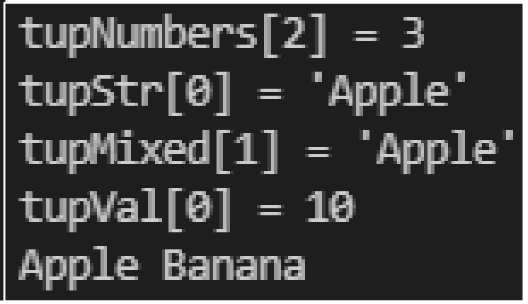
```
tupNumbers = (1, 2, 3, 4, 5)
tupStr = ('Apple', 'Banana')
tupMixed = (1, 'Apple')
tupVal = (10,)
```

8.2 讀取元組的內容

- 讀取元組內容跟讀取串列內容一樣使用[]。



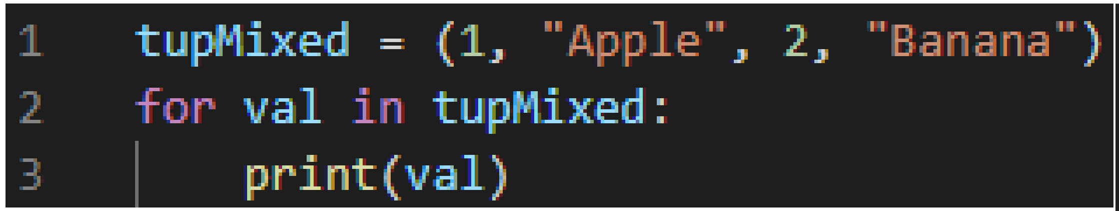
```python
tupNumbers = (1, 2, 3, 4, 5)
tupStr = ("Apple", "Banana")
tupMixed = (1, "Apple")
tupVal = (10, )
print(f"{tupNumbers[2] = }")
print(f"{tupStr[0] = }")
print(f"{tupMixed[1] = }")
print(f"{tupVal[0] = }")

strApple, strBanana = ("Apple", "Banana")
print(strApple, strBanana)
```



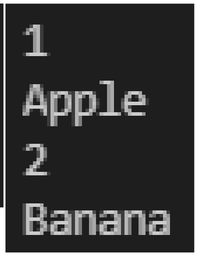
```
tupNumbers[2] = 3
tupStr[0] = 'Apple'
tupMixed[1] = 'Apple'
tupVal[0] = 10
Apple Banana
```

8.3 遍歷所有元組元素

- 可用 for 迴圈遍歷所有元組的元素。

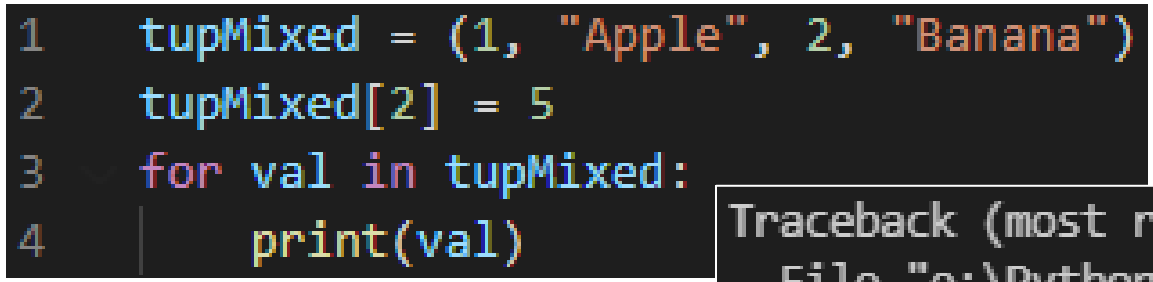


```python
tupMixed = (1, "Apple", 2, "Banana")
for val in tupMixed:
    print(val)
```

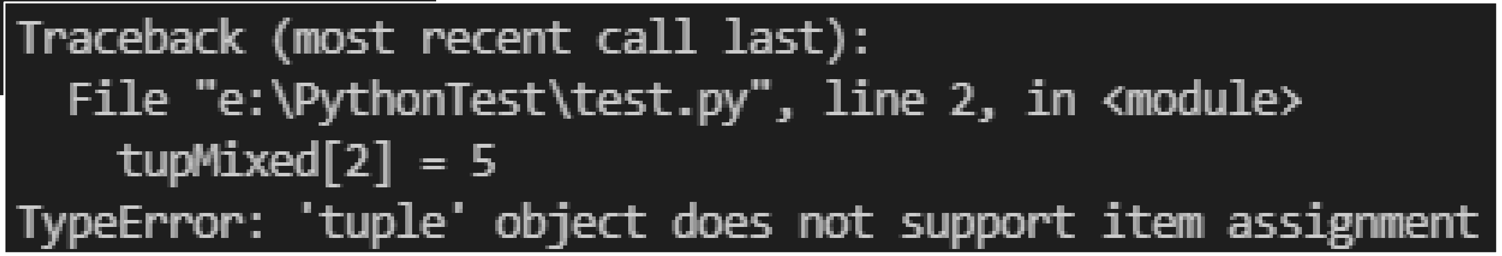


```
1
Apple
2
Banana
```

8.4 修改元組內容產生錯誤的實例

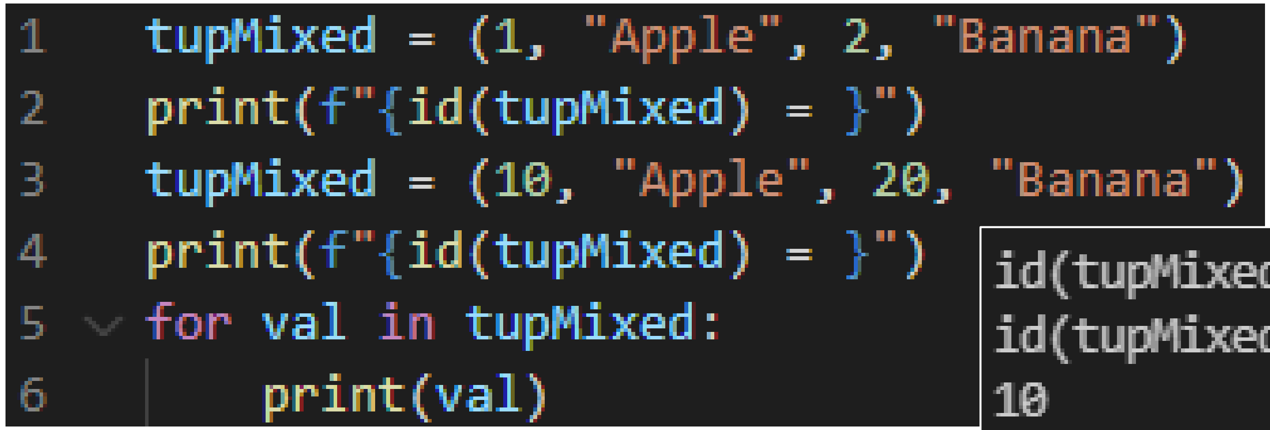


```
tupMixed = (1, "Apple", 2, "Banana")
tupMixed[2] = 5
for val in tupMixed:
    print(val)
```

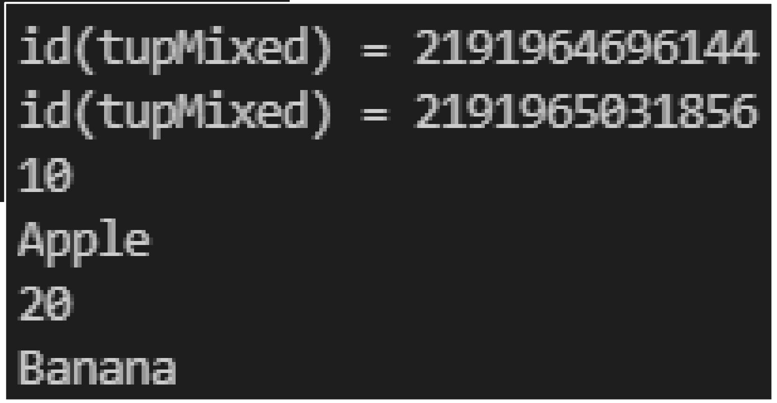


```
Traceback (most recent call last):
  File "e:\PythonTest\test.py", line 2, in <module>
    tupMixed[2] = 5
TypeError: 'tuple' object does not support item assignment
```

8.5 可以使用全新定義方式修改元組元素

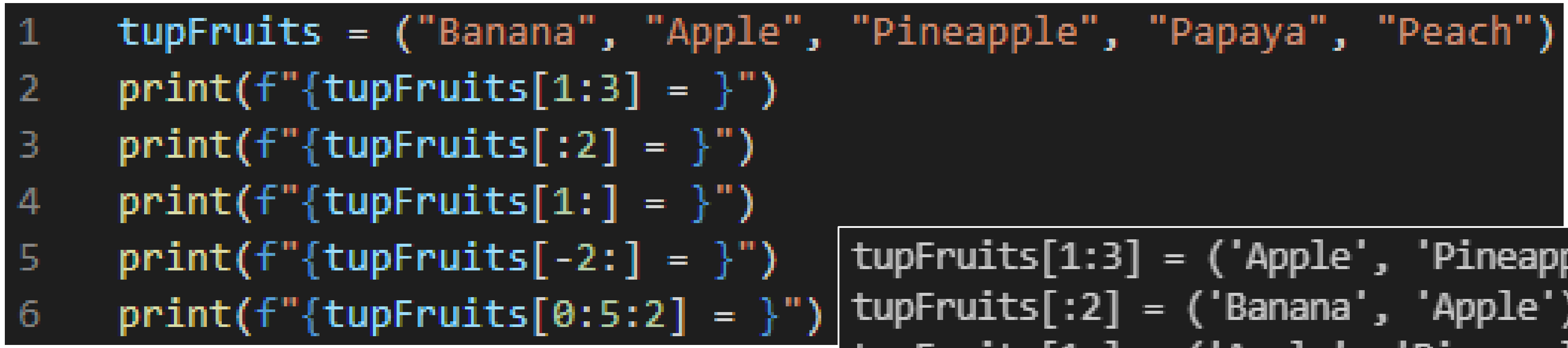


```python
tupMixed = (1, "Apple", 2, "Banana")
print(f"{id(tupMixed) = }")
tupMixed = (10, "Apple", 20, "Banana")
print(f"{id(tupMixed) = }")
for val in tupMixed:
    print(val)
```

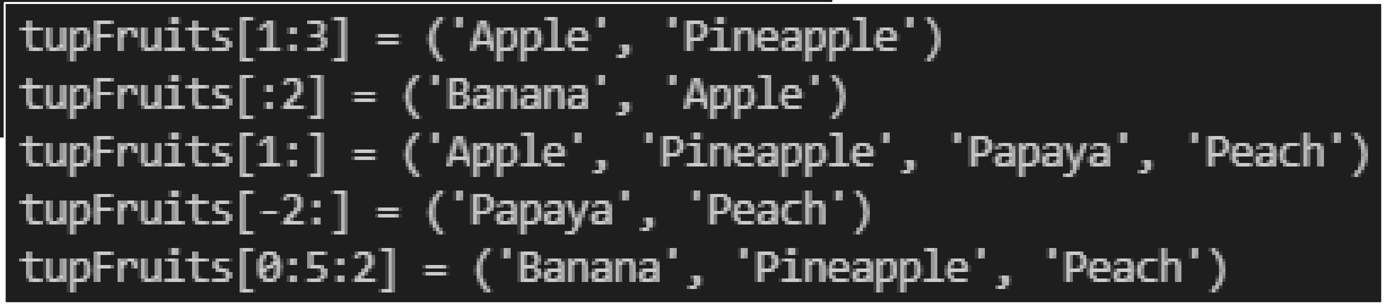


```
id(tupMixed) = 2191964696144
id(tupMixed) = 2191965031856
10
Apple
20
Banana
```

8.6 元組切片 (tuple slices)



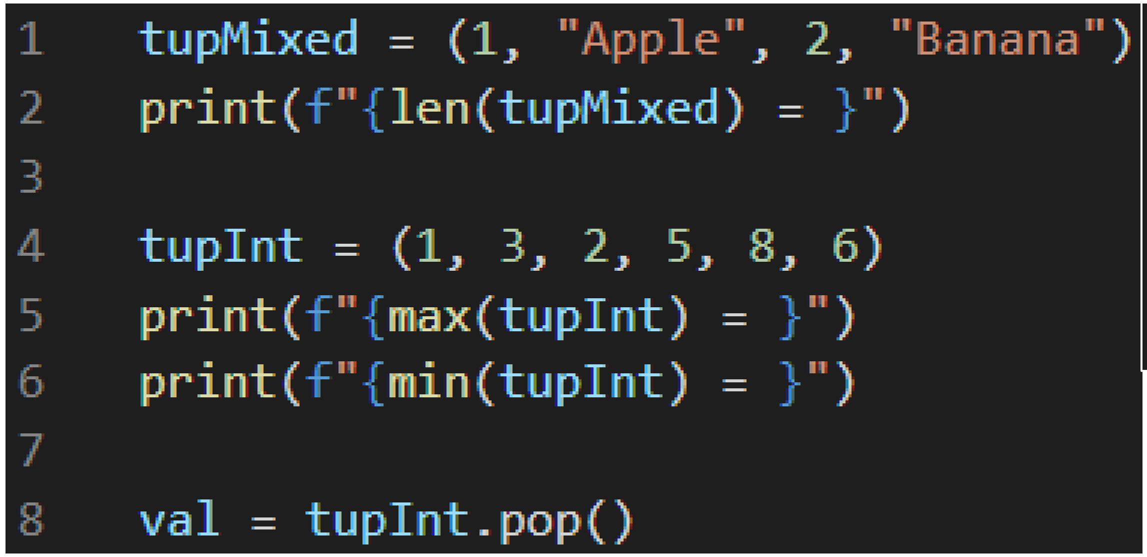
```python
tupFruits = ("Banana", "Apple", "Pineapple", "Papaya", "Peach")
print(f"{tupFruits[1:3] = }")
print(f"{tupFruits[:2] = }")
print(f"{tupFruits[1:] = }")
print(f"{tupFruits[-2:] = }")
print(f"{tupFruits[0:5:2] = }")
```



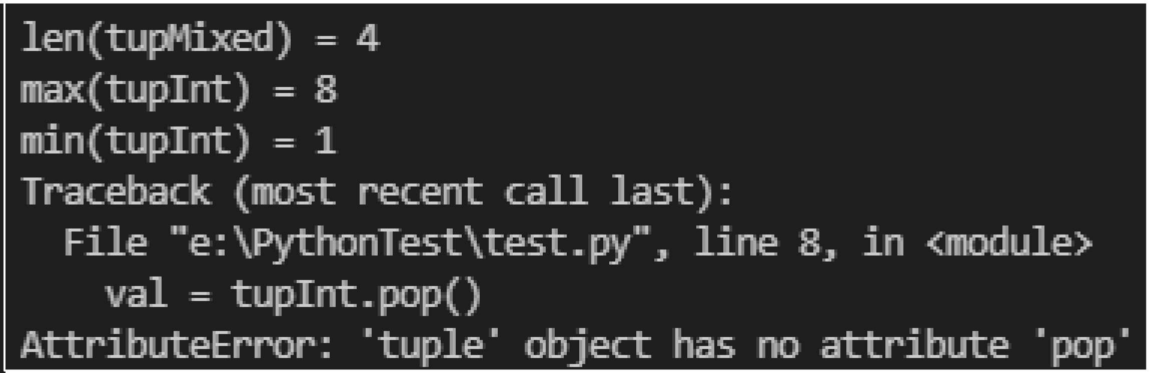
```
tupFruits[1:3] = ('Apple', 'Pineapple')
tupFruits[:2] = ('Banana', 'Apple')
tupFruits[1:] = ('Apple', 'Pineapple', 'Papaya', 'Peach')
tupFruits[-2:] = ('Papaya', 'Peach')
tupFruits[0:5:2] = ('Banana', 'Pineapple', 'Peach')
```

8.7 方法與函數

- 若是應用在串列上的方法或函數不會更改元素內容，則可以用在元組上。如 len(), max(), min()
- 若是應用在串列上的方法或函數會更改元素內容，則不可以用在元組上。如 append(), insert(), pop(), …



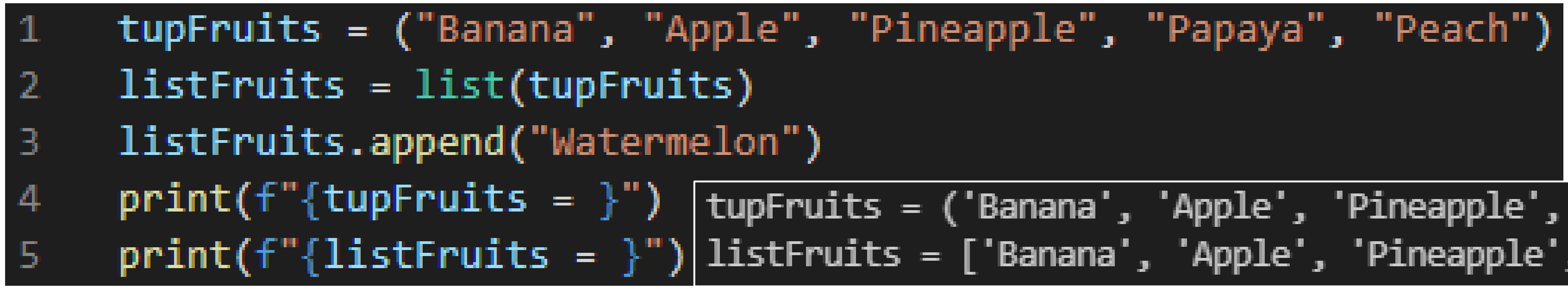
```python
tupMixed = (1, "Apple", 2, "Banana")
print(f"{len(tupMixed) = }")

tupInt = (1, 3, 2, 5, 8, 6)
print(f"{max(tupInt) = }")
print(f"{min(tupInt) = }")

val = tupInt.pop()
```



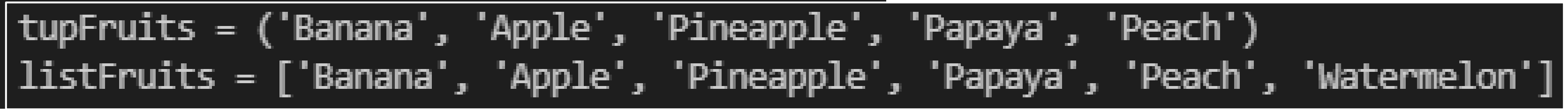
```
len(tupMixed) = 4
max(tupInt) = 8
min(tupInt) = 1
Traceback (most recent call last):
  File "e:\PythonTest\test.py", line 8, in <module>
    val = tupInt.pop()
AttributeError: 'tuple' object has no attribute 'pop'
```

8.8 串列與元組互換

- list(元組)：將元組資料型態改為串列

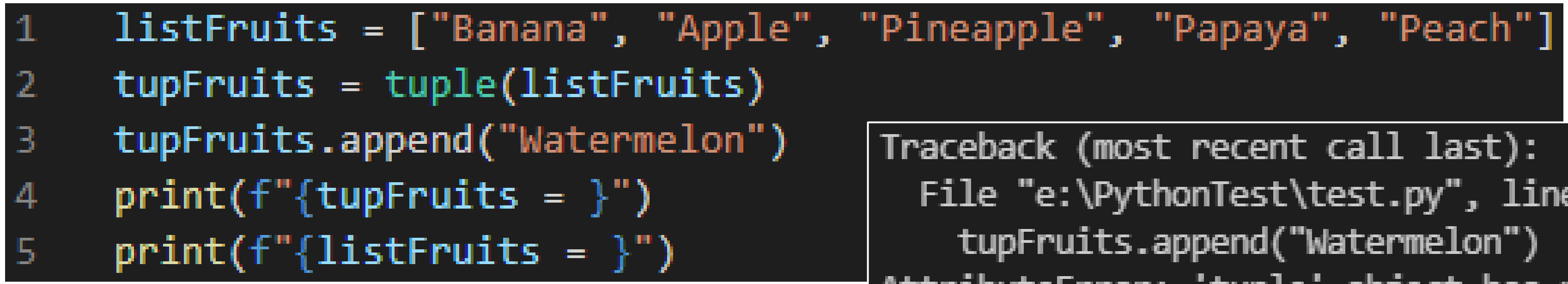


```python
tupFruits = ("Banana", "Apple", "Pineapple", "Papaya", "Peach")
listFruits = list(tupFruits)
listFruits.append("Watermelon")
print(f"{tupFruits = }")
print(f"{listFruits = }")
```

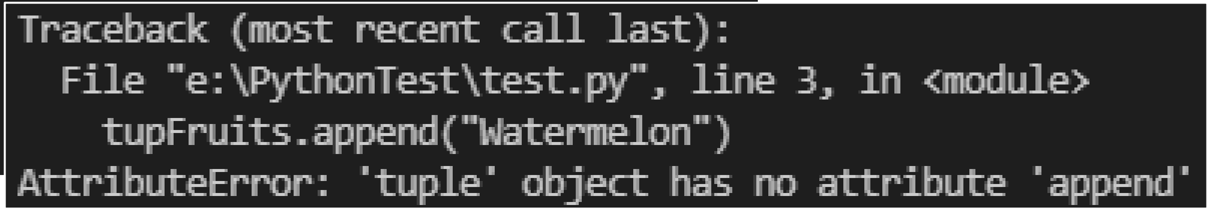


```
tupFruits = ('Banana', 'Apple', 'Pineapple', 'Papaya', 'Peach')
listFruits = ['Banana', 'Apple', 'Pineapple', 'Papaya', 'Peach', 'Watermelon']
```

- tuple(串列)：將串列資料型態改為元組



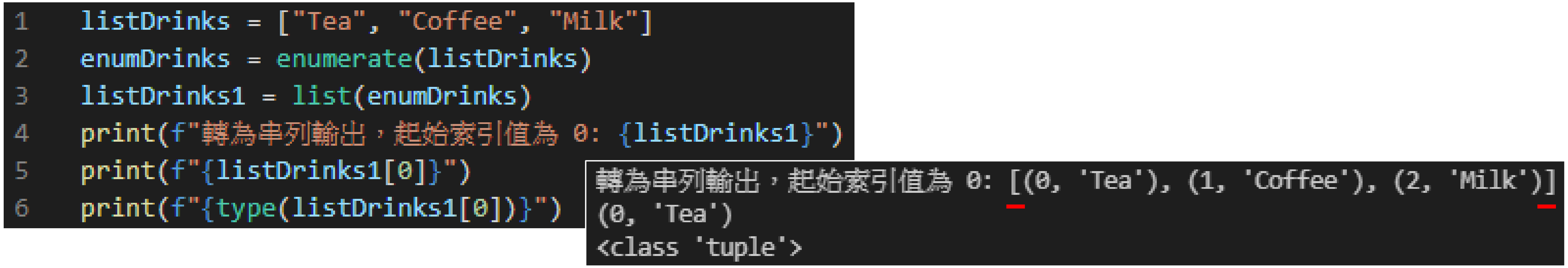
```python
listFruits = ["Banana", "Apple", "Pineapple", "Papaya", "Peach"]
tupFruits = tuple(listFruits)
tupFruits.append("Watermelon")
print(f"{tupFruits = }")
print(f"{listFruits = }")
```



```
Traceback (most recent call last):
  File "e:\PythonTest\test.py", line 3, in <module>
    tupFruits.append("Watermelon")
AttributeError: 'tuple' object has no attribute 'append'
```

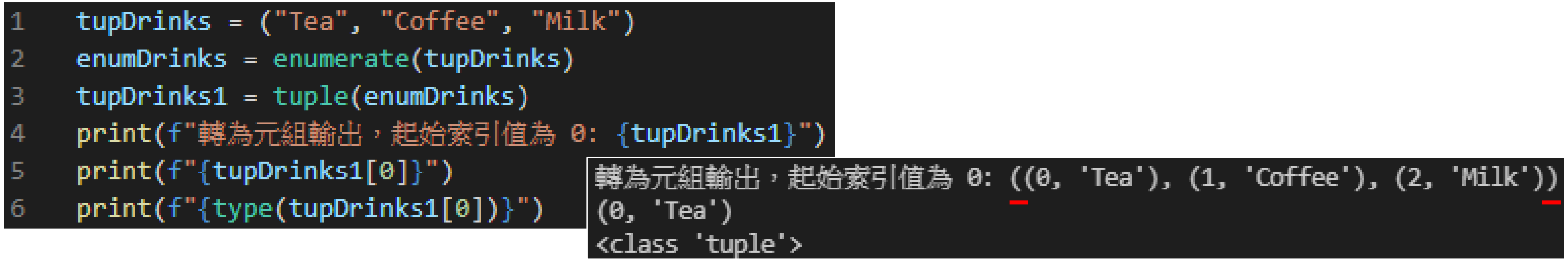
8.9 enumerate 物件使用在元組

- 當我們將用 enumerate() 函數產生的物件轉回串列時，串列內的配對元素皆為元組。



```python
listDrinks = ["Tea", "Coffee", "Milk"]
enumDrinks = enumerate(listDrinks)
listDrinks1 = list(enumDrinks)
print(f"轉為串列輸出，起始索引值為 0: {listDrinks1}")
print(f"{listDrinks1[0]}")
print(f"{type(listDrinks1[0])}")
```

```
轉為串列輸出，起始索引值為 0: [(0, 'Tea'), (1, 'Coffee'), (2, 'Milk')]
(0, 'Tea')
<class 'tuple'>
```

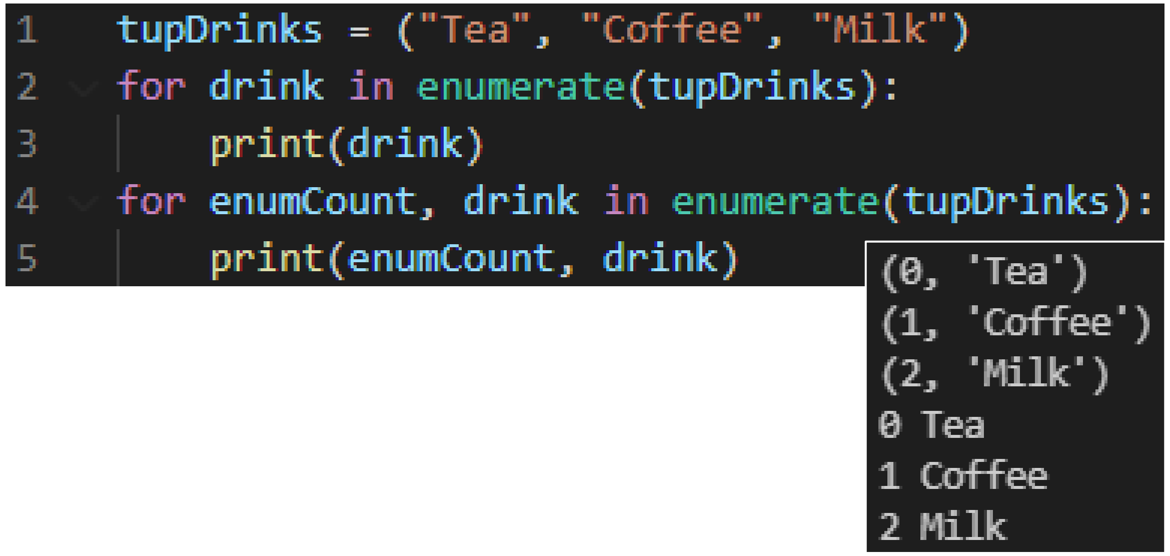


```python
tupDrinks = ("Tea", "Coffee", "Milk")
enumDrinks = enumerate(tupDrinks)
tupDrinks1 = tuple(enumDrinks)
print(f"轉為元組輸出，起始索引值為 0: {tupDrinks1}")
print(f"{tupDrinks1[0]}")
print(f"{type(tupDrinks1[0])}")
```

```
轉為元組輸出，起始索引值為 0: ((0, 'Tea'), (1, 'Coffee'), (2, 'Milk'))
(0, 'Tea')
<class 'tuple'>
```

8.9 enumerate 物件使用在元組

- 當我們將用 enumerate() 函數產生的物件轉回串列時，串列內的配對元素皆為元組。

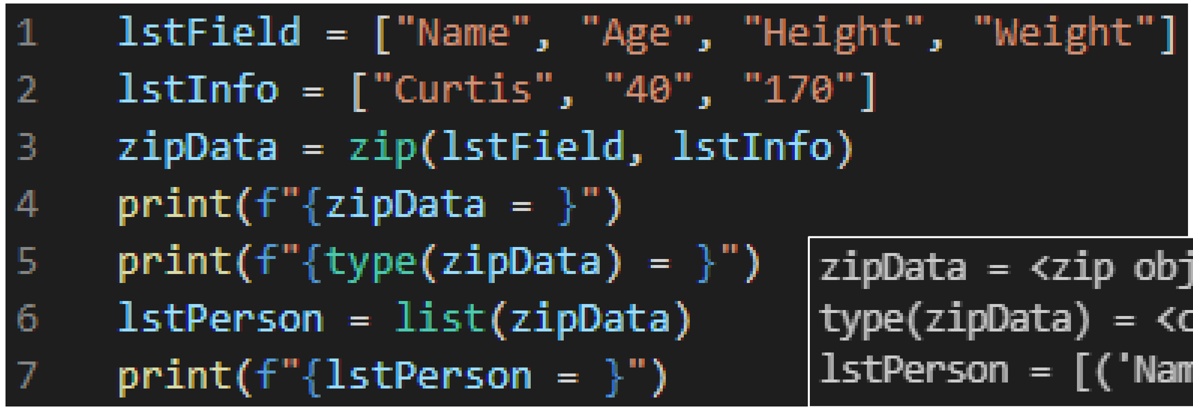


```python
tupDrinks = ("Tea", "Coffee", "Milk")
for drink in enumerate(tupDrinks):
    print(drink)
for enumCount, drink in enumerate(tupDrinks):
    print(enumCount, drink)
```

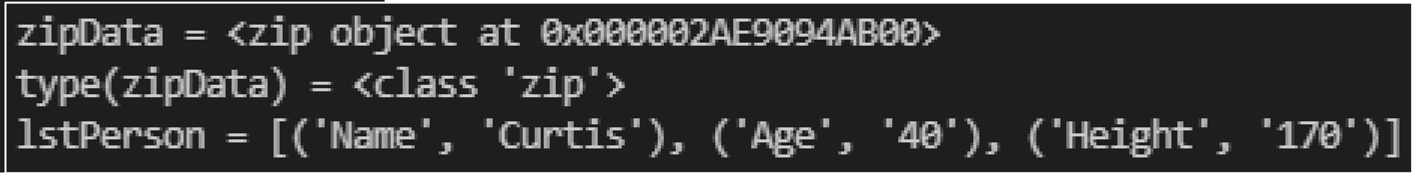
```
(0, 'Tea')
(1, 'Coffee')
(2, 'Milk')
0 Tea
1 Coffee
2 Milk
```

8.10 使用zip( )打包多個物件

- zip()將多個可迭代物件打包成zip物件，然後未來視需要將此zip物件轉成串列(使用list( ))或其它物件。



```python
lstField = ["Name", "Age", "Height", "Weight"]
lstInfo = ["Curtis", "40", "170"]
zipData = zip(lstField, lstInfo)
print(f"{zipData = }")
print(f"{type(zipData) = }")
lstPerson = list(zipData)
print(f"{lstPerson = }")
```

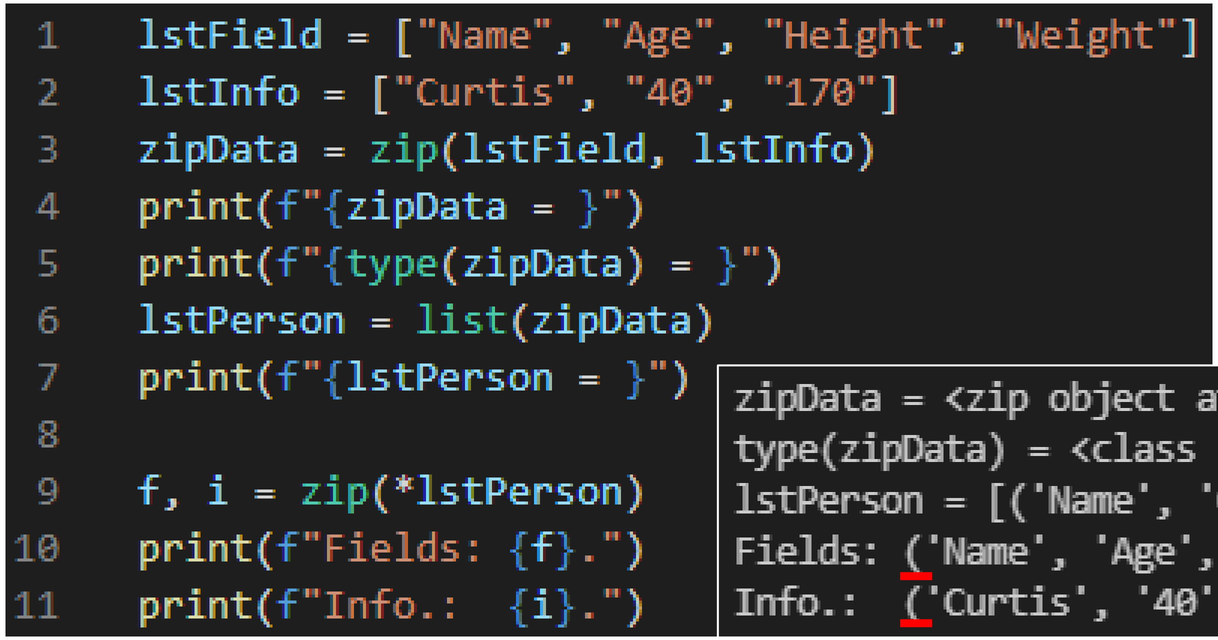


```
zipData = <zip object at 0x000002AE9094AB00>
type(zipData) = <class 'zip'>
lstPerson = [('Name', 'Curtis'), ('Age', '40'), ('Height', '170')]
```

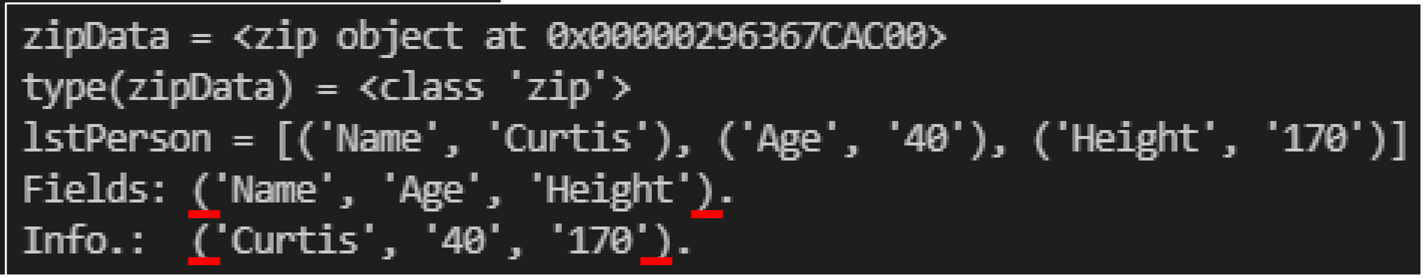
  - zip() 函數內的參數，資料長度不相等時，會以短的為主。

8.10 使用zip( )打包多個物件

- 在zip()函數中，參數前多加個"*"，相當於 unzip串列。



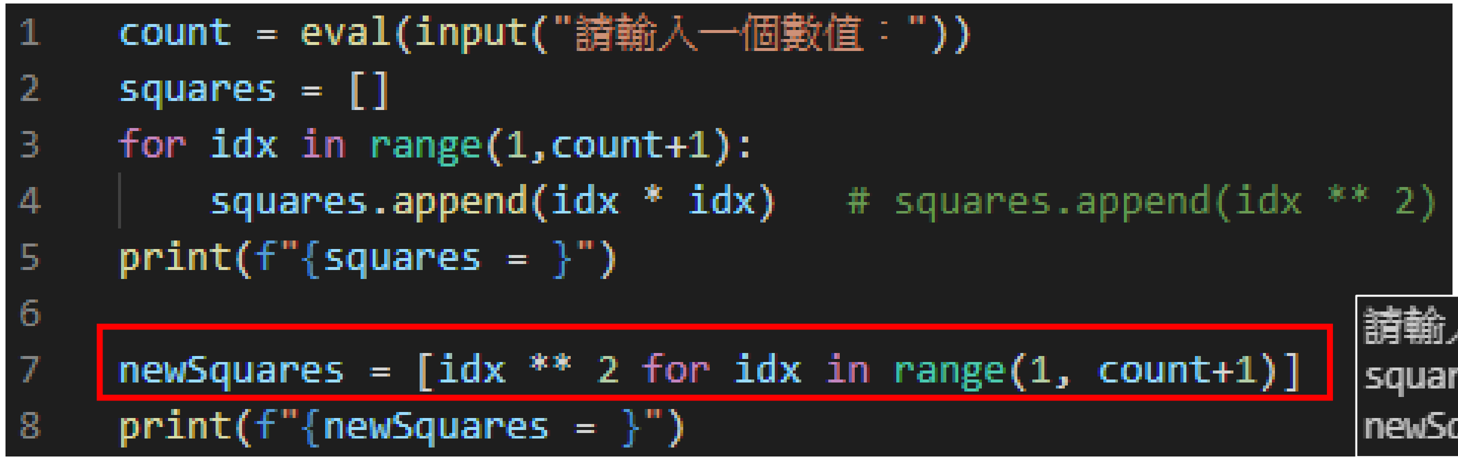
```python
lstField = ["Name", "Age", "Height", "Weight"]
lstInfo = ["Curtis", "40", "170"]
zipData = zip(lstField, lstInfo)
print(f"{zipData = }")
print(f"{type(zipData) = }")
lstPerson = list(zipData)
print(f"{lstPerson = }")

f, i = zip(*lstPerson)
print(f"Fields: {f}.")
print(f"Info.:  {i}.")
```



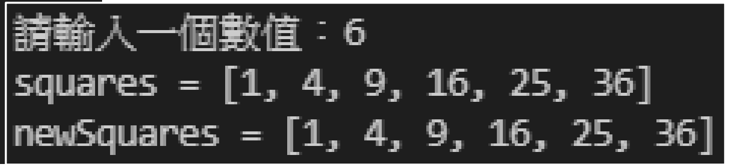
```
zipData = <zip object at 0x00000296367CAC00>
type(zipData) = <class 'zip'>
lstPerson = [('Name', 'Curtis'), ('Age', '40'), ('Height', '170')]
Fields: ('Name', 'Age', 'Height').
Info.:  ('Curtis', '40', '170').
```

8.11 生成式 (generator)

- Chapter 7: List generator

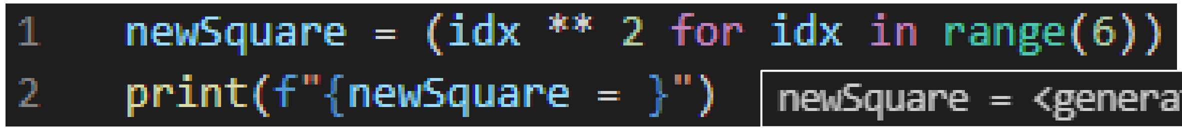


```python
count = eval(input("請輸入一個數值："))
squares = []
for idx in range(1,count+1):
    squares.append(idx * idx)    # squares.append(idx ** 2)
print(f"{squares = }")

newSquares = [idx ** 2 for idx in range(1, count+1)]
print(f"{newSquares = }")
```

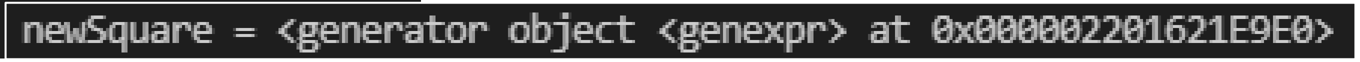


```
請輸入一個數值：6
squares = [1, 4, 9, 16, 25, 36]
newSquares = [1, 4, 9, 16, 25, 36]
```

- 元組的生成式
  - 新元組 = (運算式 for 運算式 in 可迭代項目)
  - num = (n for n in range(6))



```python
newSquare = (idx ** 2 for idx in range(6))
print(f"{newSquare = }")
```



```
newSquare = <generator object <genexpr> at 0x000002201621E9E0>
```

8.11 生成式 (generator)

- 生成式(generator)物件：

```python
newSquare = (idx ** 2 for idx in range(6))
print(f"{newSquare = }")
print(f"{type(newSquare)}")
```

```
newSquare = <generator object <genexpr> at 0x000002B676F7E9E0>
<class 'generator'>
```

  - 可迭代物件。

```python
square = (idx ** 2 for idx in range(6))
for val in square:
    print(val, end=" ")
```

```
0 1 4 9 16 25
```

  - 可用 list()/tuple() 將此生成式變成串列/元組。
  - 只能讀取一次，第二次將得到空串列/元組。

```python
square = (idx ** 2 for idx in range(6))
lstSquare = list(square)
print(lstSquare)
tupSquare = tuple(square)
print(tupSquare)
```

```
[0, 1, 4, 9, 16, 25]
()
```

```python
square = (idx ** 2 for idx in range(6))
tupSquare = tuple(square)
print(tupSquare)
lstSquare = list(square)
print(lstSquare)
```

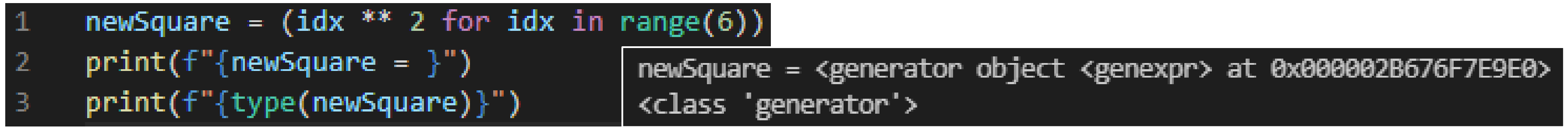
```
(0, 1, 4, 9, 16, 25)
[]
```

```python
square = (idx ** 2 for idx in range(6))
for val in square:
    print(val, end=" ")
print()
tupSquare = tuple(square)
print(tupSquare)
lstSquare = list(square)
print(lstSquare)
```

```
0 1 4 9 16 25
()
[]
```

8.11 生成式 (generator)

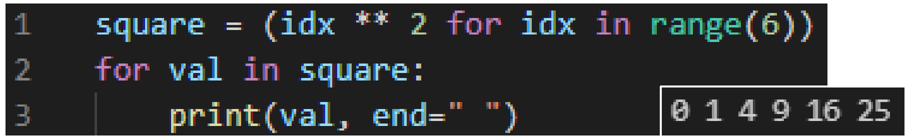
- 生成式(generator)物件：



```python
newSquare = (idx ** 2 for idx in range(6))
print(f"{newSquare = }")
print(f"{type(newSquare)}")
```

```
newSquare = <generator object <genexpr> at 0x000002B676F7E9E0>
<class 'generator'>
```

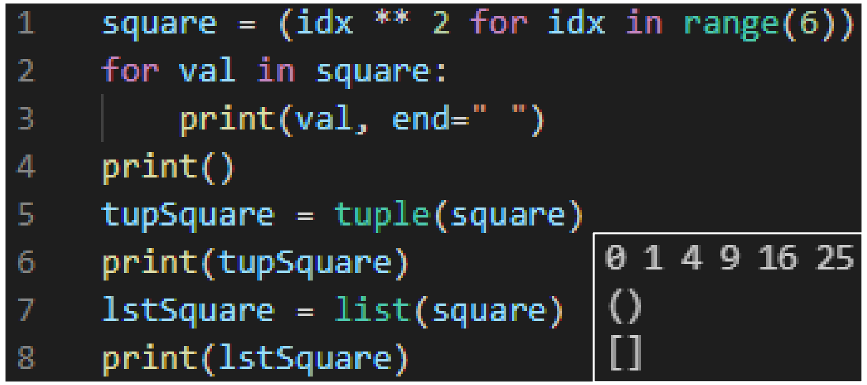
  - 可迭代物件。



```python
square = (idx ** 2 for idx in range(6))
for val in square:
    print(val, end=" ")
```

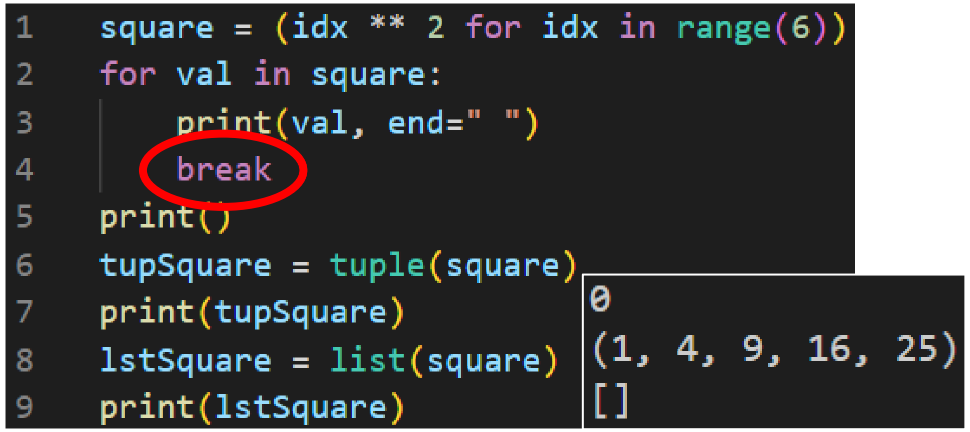
```
0 1 4 9 16 25
```

  - 可用 list()/tuple() 將此生成式變成串列/元組。
  - 只能讀取一次，第二次將得到空串列/元組。



```python
square = (idx ** 2 for idx in range(6))
for val in square:
    print(val, end=" ")
print()
tupSquare = tuple(square)
print(tupSquare)
lstSquare = list(square)
print(lstSquare)
```

```
0 1 4 9 16 25
()
[]
```

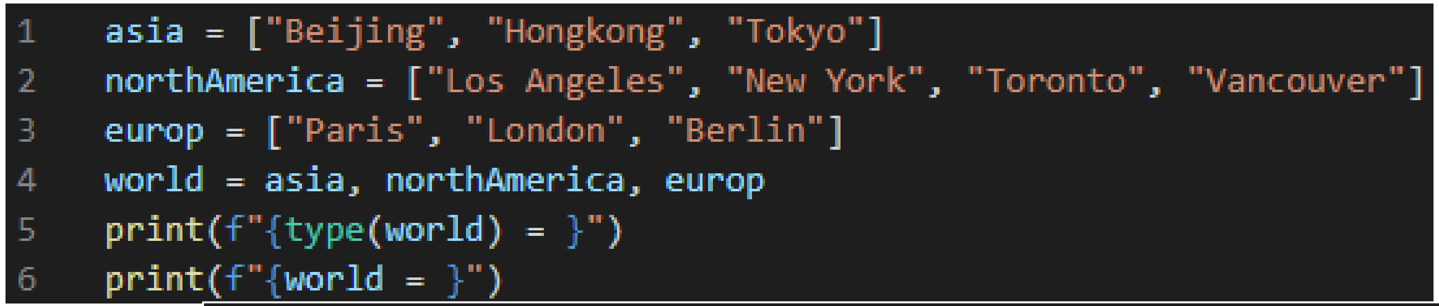


```python
square = (idx ** 2 for idx in range(6))
for val in square:
    print(val, end=" ")
    break
print()
tupSquare = tuple(square)
print(tupSquare)
lstSquare = list(square)
print(lstSquare)
```

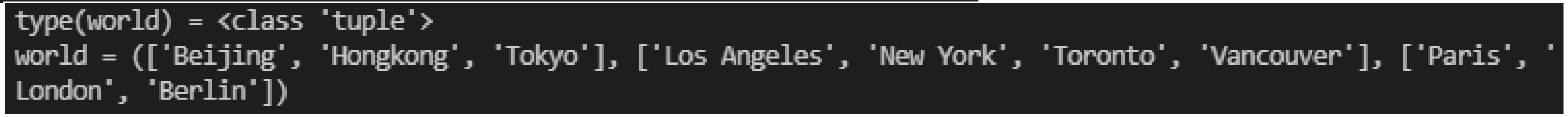
```
0
(1, 4, 9, 16, 25)
[]
```

8.12 製作大型的元組資料

- 元組的內容是串列



```python
asia = ["Beijing", "Hongkong", "Tokyo"]
northAmerica = ["Los Angeles", "New York", "Toronto", "Vancouver"]
europ = ["Paris", "London", "Berlin"]
world = asia, northAmerica, europ
print(f"{type(world) = }")
print(f"{world = }")
```



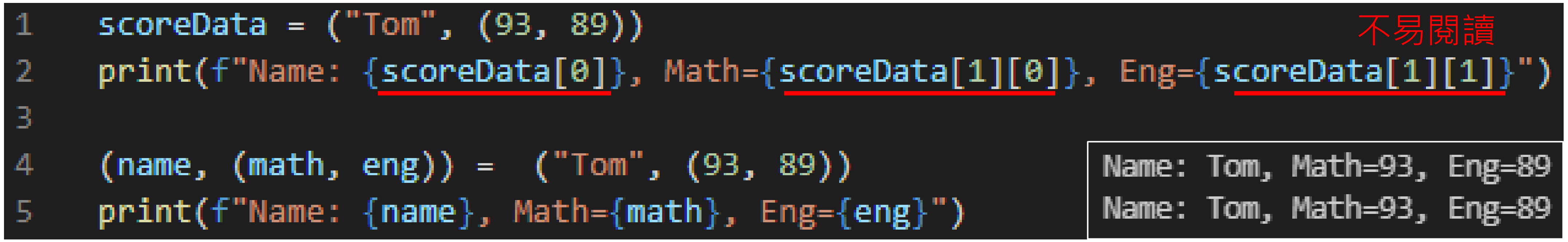
```
type(world) = <class 'tuple'>
world = (['Beijing', 'Hongkong', 'Tokyo'], ['Los Angeles', 'New York', 'Toronto', 'Vancouver'], ['Paris', '
London', 'Berlin'])
```

8.13 元組的功能

- 可以更安全的保護資料
  - 資料無法修改，可以安全的被保護。
  - 例如：影像處理時的影像寬高。
- 增加程式執行速度
  - 元組結構比串列簡單，佔用較少系統資源。

8.14 多重指定、打包與解包

- 串列、元組、字典、集合...統稱為容器。
- 在多重指定中，等號左右兩邊也可以為容器，只要結構相同即可。
    - x, y = (10, 20) ➔ 元組解包 (tuple unpacking)，將元素設定給對應的變數。
    - a, b, *c = 1, 2, 3, 4, 5 ➔ c = [3, 4, 5]，稱為將資料打包 (packing) 成串列給c。
    - [a, b, c] = (1, 2, 3) ➔ 將兩邊容器的資料都 unpacking，做多重指定。
    - [a, [b, c]] = (1, (2, 3)) ➔ 多維度資料也可以 unpacking，做多重指定。



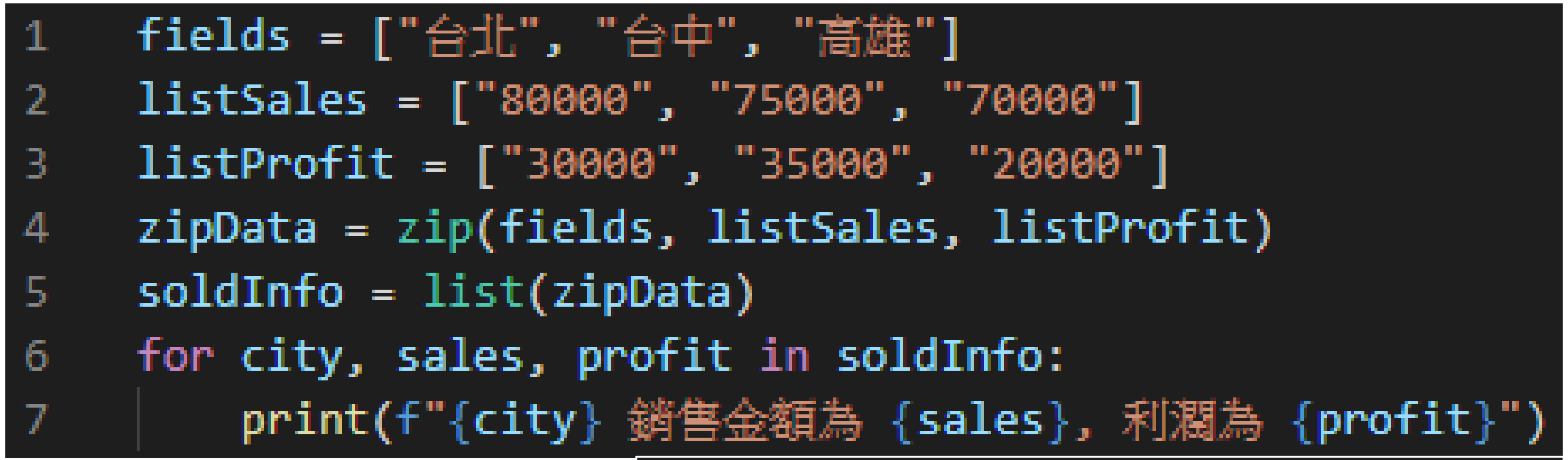
```python
scoreData = ("Tom", (93, 89))
print(f"Name: {scoreData[0]}, Math={scoreData[1][0]}, Eng={scoreData[1][1]}")

(name, (math, eng)) =  ("Tom", (93, 89))
print(f"Name: {name}, Math={math}, Eng={eng}")
```

不易閱讀

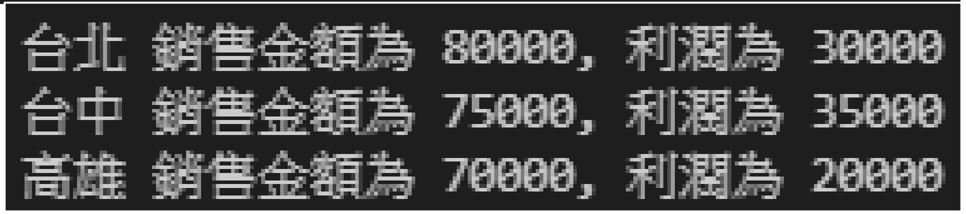
```
Name: Tom, Math=93, Eng=89
Name: Tom, Math=93, Eng=89
```

8.14 多重指定、打包與解包

- 用 for 迴圈解包。



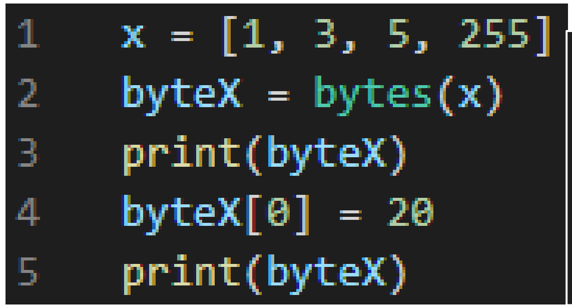
```python
fields = ["台北", "台中", "高雄"]
listSales = ["80000", "75000", "70000"]
listProfit = ["30000", "35000", "20000"]
zipData = zip(fields, listSales, listProfit)
soldInfo = list(zipData)
for city, sales, profit in soldInfo:
    print(f"{city} 銷售金額為 {sales}, 利潤為 {profit}")
```



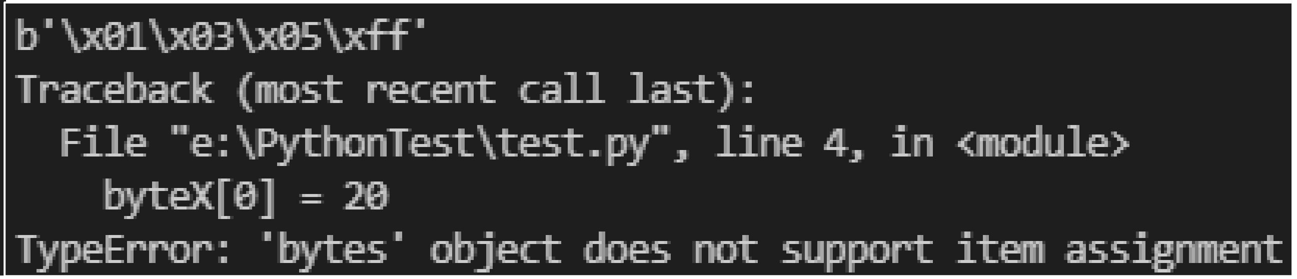
```
台北 銷售金額為 80000, 利潤為 30000
台中 銷售金額為 75000, 利潤為 35000
高雄 銷售金額為 70000, 利潤為 20000
```

8.15 再談 bytes 與 bytearray

- Chapter 3 講到，可將字串轉為 bytes 資料。這是種二進制的資料格式，總共有兩種：
  - bytes: 內容是不可變的，可想像為元組。可用 bytes() 將資料轉成 bytes 資料。

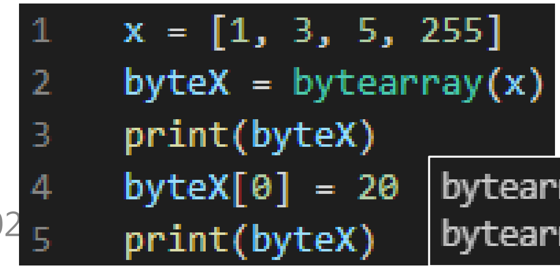


```python
x = [1, 3, 5, 255]
byteX = bytes(x)
print(byteX)
byteX[0] = 20
print(byteX)
```

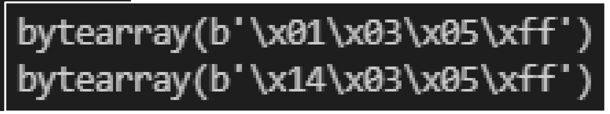


```
b'\x01\x03\x05\xff'
Traceback (most recent call last):
  File "e:\PythonTest\test.py", line 4, in <module>
    byteX[0] = 20
TypeError: 'bytes' object does not support item assignment
```

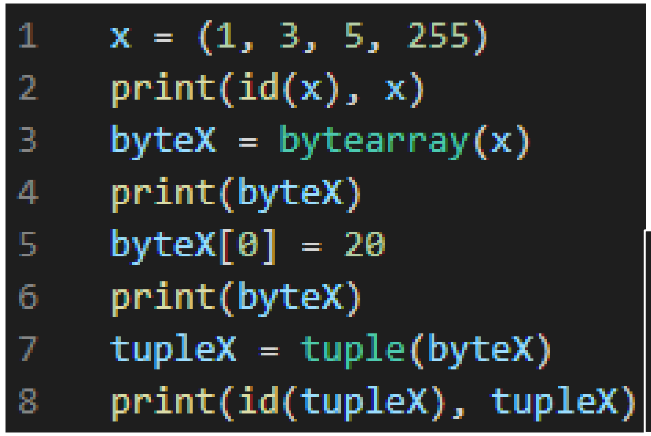
  - bytearray:內容是可變的，可想像為串列。可用 bytearray() 將資料轉成 bytearray 資料。



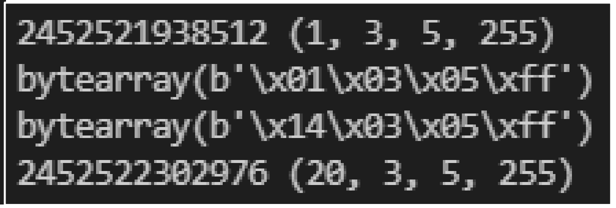
```python
x = [1, 3, 5, 255]
byteX = bytearray(x)
print(byteX)
byteX[0] = 20
print(byteX)
```



```
bytearray(b'\x01\x03\x05\xff')
bytearray(b'\x14\x03\x05\xff')
```



```python
x = (1, 3, 5, 255)
print(id(x), x)
byteX = bytearray(x)
print(byteX)
byteX[0] = 20
print(byteX)
tupleX = tuple(byteX)
print(id(tupleX), tupleX)
```



```
2452521938512 (1, 3, 5, 255)
bytearray(b'\x01\x03\x05\xff')
bytearray(b'\x14\x03\x05\xff')
2452522302976 (20, 3, 5, 255)
```

8.14 動手練習

- 使用 divmod() 函數計算一年 365 天是幾個星期又幾天。
  - 商, 餘數 = divmod(被除數, 除數)
- 使用元組儲存資料後做統計：計算 5, 6, 8, 9, 13 的平均值、變異數與標準差。

平均值：

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i = \frac{x_1 + x_2 + \cdots + x_n}{n}$$

變異數：

$$\text{variance} = \frac{1}{n}\sum_{i=1}^{n} (x_i - \bar{x})^2$$

標準差：

$$\text{standard deviation} = \sqrt{\frac{1}{n}\sum_{i=1}^{n} (x_i - \bar{x})^2}$$