


# Python MTA



玩中學

做中學

錯中學

# 變數命名規則

- 不可含有空白
- 大小寫有區別
- 不可以用數字開頭
- 數字用在中間或後面
- 不要太長，看到名稱知道意思就好

# 建立變數

語法

變數名稱 = 變數值

練習

設定變數 `pi` 為 `3.14`

```
>>> pi = 3.14
>>> print(pi)
3.14
>>>
```

玩中學

做中學

錯中學

# 修改變數

變數可以被重新設定

語法

變數名稱 = 變數值

練習

重新設定變數 `pi` 為 `3.14159`

```
pi = 3.14
print(pi)
pi = 3.14159
print(pi)
```

```
>>> pi = 3.14
>>> print(pi)
3.14
>>> pi = 3.14159
>>> print(pi)
3.14159
```

玩中學

做中學

錯中學

# 刪除變數

變數可以被重新設定

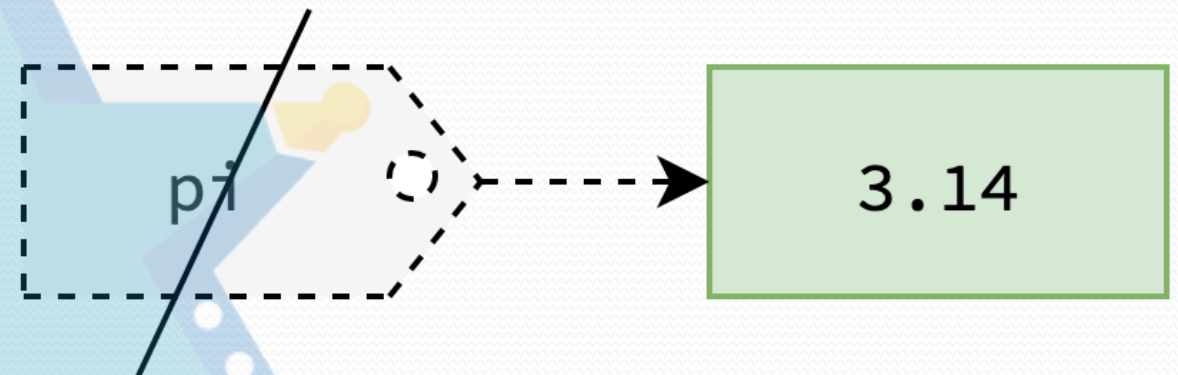
語法

```
del 變數名稱
```

練習

刪除變數 `pi`

```
pi = 3.14  
del pi  
print(pi)
```



```
>>> pi = 3.14  
>>> del pi  
>>> print(pi)  
Traceback (most recent call  
last):  
  File "<stdin>", line 1, in  
  <module>  
NameError: name 'pi' is not  
defined
```



# 格式化顯示數字

符號	說明
%d	以 10 進位整數方式輸出
%3d	3 個字元寬度顯示整數，不滿 3 個則補空白
%03d	以 3 個字元寬度顯示整數，不滿 3 個則補 0
%f	將浮點數以 10 進位方式輸出
%.2f	以浮點數表示，設定小數位數為 2 位
%s	使用str()將字串輸出

# 整數格式化法1

# 以整數表示

```
print('I %d savage' % 94)
```

# 以 3 個字元寬度顯示整數，不滿 3 個則補空白

```
print('I %3d savage' % 94)
```

# 以 3 個字元寬度顯示整數，不滿 3 個則補 0

```
print('I %03d savage' % 94)
```

# 整數格式化法2

以整數表示

```
print("I {0:d} savage".format(94))
```

以 3 個字元寬度顯示整數，不滿 3 個則補空白

```
print("I {0:3d} savage".format(94))
```

#以 3 個字元寬度顯示整數，不滿 3 個則補 0

```
print("I {0:03d} savage".format(94))
```



# 浮點數格式化法1

以浮點數表示，預設小數位數為 6 位

```
print('I   %f   savage' % 94)
```

以浮點數表示，設定小數位數為 2 位

```
print('I   %.2f   savage' % 94)
```

# 浮點數格式化法2

以浮點數表示，預設小數位數為 6 位

```
print('I {0:f} savage'.format(94))
```

以浮點數表示，設定小數位數為 2 位

```
print('I {0:.2f} savage'.format(94))
```

# 字串格式化法

以字串表示

```
print('I %s savage' % '94')
```

以字串表示

```
print('I {0:s} savage'.format('94'))
```

# 串列資料型態 List

- 依序儲存資料
- 可以修改元素值
- 使用中括號表示，用逗號分隔資料內容

['蘋果', '香蕉', '葡萄']

# 建立串列(List) 指令格式

使用中括號建立串列

```
[ 資料1, 資料2, ... ]
```

## 範例

- `[]`
- `['蘋果']`
- `['a', 'b']`
- `[1, 2, 3]`

```
>>> []  
[]  
>>> ['蘋果']  
['蘋果']  
>>> ['a', 'b']  
['a', 'b']  
>>> [1, 2, 3]  
[1, 2, 3]  
>>>
```



# len 函數

取得串列長度

指令格式

```
len(串列)
```

範例

- `len([])`
- `len(['蘋果'])`
- `len(['a', 'b'])`
- `len([1, 2, 3])`

```
>>> len([])
```

```
0
```

```
>>> len(['蘋果'])
```

```
1
```

```
>>> len(['a', 'b'])
```

```
2
```

```
>>> len([1, 2, 3])
```

```
3
```

# 合併串列

將兩個串列相連結，合成一個新串列

指令格式

串列1 + 串列2

範例

```
>>> [1, 2] + ['b', 'c']  
[1, 2, 'b', 'c']
```

- `[1, 2] + ['b', 'c']`

# 分割字串

將字串內依照分隔字元切割成多個字串，保存在串列中

指令格式

```
字串.split(字元)
```

範例

- `'1,2,3'.split(',')`
- `'2020/1/1'.split('/')`

```
>>> '1,2,3'.split(',')  
['1', '2', '3']
```

```
>>> '2020/1/1'.split('/')  
['2020', '1', '1']
```

# 重複串列

將串列內的元素重複整數次

指令格式

串列 \* 重複次數

範例

```
>>> [1,2] * 2  
[1, 2, 1, 2]
```

- `[1,2] * 2`

# 重組字串

將串列重組成新的字串

範例

指令格式

```
分隔字元.join(串列)
```

- `img=['1', '2', '3']`
- `','.join(img)`

```
>>> ','.join(img)  
1,2,3
```



# list 函數

將其他資料型態轉換成串列

指令格式

```
list(資料)
```

範例

- `list('abc')`
- `list([4,5,6])`
- `list(range(3))`

```
>>> list('abc')  
['a', 'b', 'c']  
>>> list([4,5,6])  
[4, 5, 6]  
>>> list(range(3))  
[0, 1, 2]
```



# 元素存取

---

玩中學

做中學

錯中學

# 取得元素

取得對應位置的元素數值

指令格式

串列[索引值]

範例

- `l = ['a', 'b', 'c']`
- `l[0]`
- `l[1]`
- `l[2]`

```
>>> l = ['a', 'b', 'c']
>>> l[0]
'a'
>>> l[1]
'b'
>>> l[2]
'c'
```

# 指定索引元素

設定對應位置的元素數值

指令格式

串列[索引值] = 數值

範例

- `l = ['a', 'b', 'c']`
- `l`
- `l[0] = 'A'`
- `l`

```
>>> l = ['a', 'b', 'c']
```

```
>>> l  
['a', 'b', 'c']
```

```
>>> l[0] = 'A'
```

```
>>> l  
['A', 'b', 'c']
```

# 取得子串列

從原本的串列中，取得範圍中的元素，生成子串列

指令格式

串列[起始值:中止值]

## 範例

- `l = [0, 1, 2, 3, 4]`
- `l[0:3]`
- `l[3:5]`
- `l[-3:]`

```
>>> l = [0, 1, 2, 3, 4]
```

```
>>> l[0:3]  
[0, 1, 2]
```

```
>>> l[3:5]  
[3, 4]
```

```
>>> l[-3:]  
[2, 3, 4]
```



# 想想看

a 是多少？

```
a = [1,2,3]
```

```
b = a
```

```
b[0] = 2
```

```
print(a)
```

玩中學

做中學

錯中學

# 想想看

a 是多少？

是 [2, 2, 3]

```
a = [1, 2, 3]
```

```
b = a
```

```
b[0] = 2
```

```
print(a)
```

# 串列方法

玩中學

做中學

錯中學

# append 方法

將元素增加到串列的最後

指令格式

```
串列.append(數值)
```

範例

```
l = [1, 2, 3]  
l.append(4)  
print(l)
```

執行結果

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

# remove 方法

移除串列中的元素

- 指定的元素必須存在
- 移除第一個出現的元素

範例

```
l = ['a', 'b', 'c', 'a']  
l.remove('a')  
print(l)
```

指令格式

```
串列.remove(元素值)
```

執行結果

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



# Del 方法

移除串列中的元素

- 指定的元素必須存在
- 移除第一個出現的元素

範例

```
l = ['a', 'b', 'c', 'a']  
del l[0]  
print(l)
```

指令格式

```
Del 串列[index]
```

執行結果

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

# insert 方法

將元素插入指定的位置

指令格式

```
串列.insert(索引位置, 數值)
```

範例

```
l = [1, 2, 3]
l.insert(0, 'A')
print(l)
```

執行結果

```
['A', 1, 2, 3]
```

# pop 方法

移除串列裡最後一個元素

指令格式

串列.pop()

範例

```
l = [1, 2, 3]
l.pop()
print(l)
```

執行結果

[1, 2]

# pop 方法

## 移除指定索引的元素

指令格式

```
串列.pop(索引)
```

範例

```
l = [1, 2, 3]  
l.pop(0)  
print(l)
```

執行結果

```
[2, 3]
```

# sort 方法

由小至大排序串列元素

指令格式

```
串列.sort()
```

範例

```
l = [3, 1, 5, 4, 2]  
l.sort()  
print(l)
```

執行結果

```
[1, 2, 3, 4, 5]
```



# reverse 方法

顛倒串列的順序

指令格式

```
串列.reverse()
```

範例

```
l = [1, 2, 3]  
l.reverse()  
print(l)
```

執行結果

```
[3, 2, 1]
```

# index 方法

找到串列中對應元素值的位置索引

指令格式

```
串列.index(元素值)
```

- 指定的元素必須存在
- 索引對應第一個找到的元素位置

範例

```
l = ['a', 'b', 'c', 'a']  
index = l.index('a')  
print(index)
```

執行結果

0