



# SCU LawTech

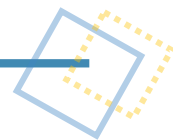
## Class #01

# Content

---

- 安裝 Anaconda
- Line 群組
- Github 帳號申請
- 變數
- 資料型態

# 安裝 Anaconda



## Anaconda Installers

Windows 

Python 3.8

64-Bit Graphical Installer (466 MB)

32-Bit Graphical Installer (397 MB)

MacOS 

Python 3.8

64-Bit Graphical Installer (462 MB)

64-Bit Command Line Installer (454 MB)

Linux 

Python 3.8

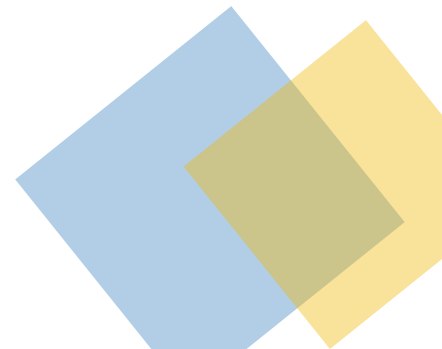
64-Bit (x86) Installer (550 MB)

64-Bit (Power8 and Power9) Installer (290 MB)

<https://www.anaconda.com/products/individual>

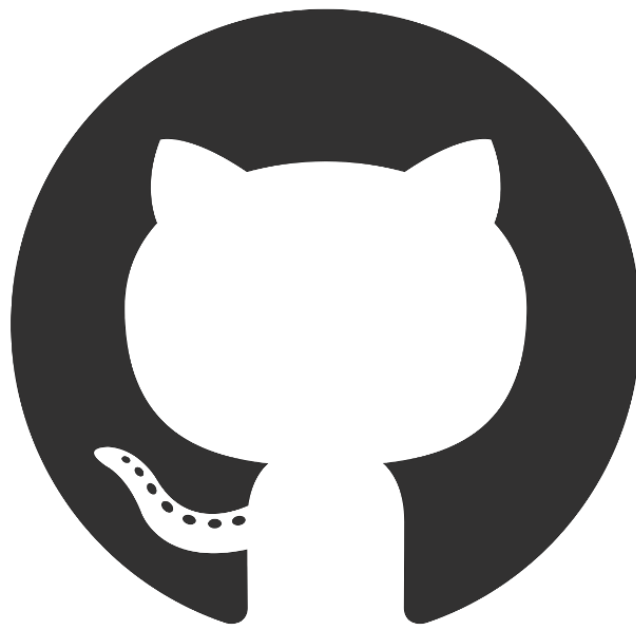
# Line 群組

---

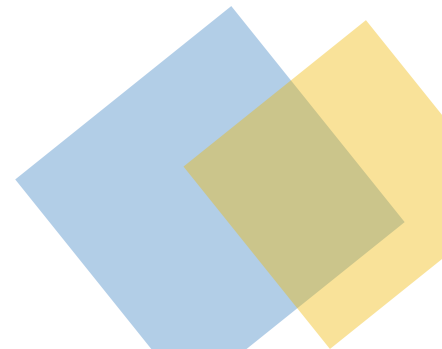


# GitHub 帳號申請

---



[https://docs.google.com/forms/d/e/1FAIpQLScZ4ktM0EpFNIZJUHfbwS7d2U7eygA-QGnhQvAL71vTKEOaYw/viewform?usp=sf\\_link](https://docs.google.com/forms/d/e/1FAIpQLScZ4ktM0EpFNIZJUHfbwS7d2U7eygA-QGnhQvAL71vTKEOaYw/viewform?usp=sf_link)



# 變數 ( Variable )

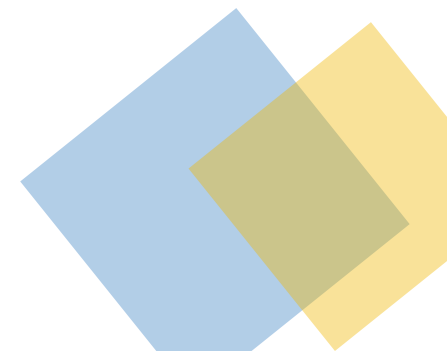
---

- 變數是為了方便在程式中調用記憶體的值
- " = " : 賦值、指派

```
In [8]: a = 123
```

```
In [9]: print(a)
```

```
123
```



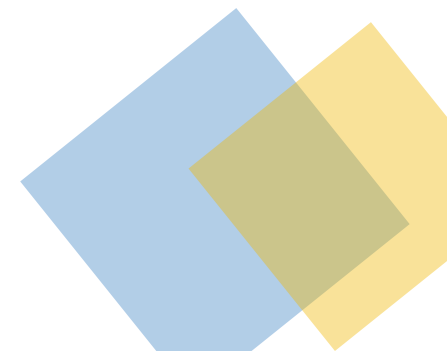
# 變數命名規則

---

- 變數名稱只能包括：大小寫字母、中文、數字、底線
- 命名時請避開「系統保留字」或「函數名稱」

<b>and</b>	<b>del</b>	<b>for</b>	<b>is</b>	<b>raise</b>
<b>assert</b>	<b>elif</b>	<b>from</b>	<b>lambda</b>	<b>return</b>
<b>break</b>	<b>else</b>	<b>global</b>	<b>not</b>	<b>try</b>
<b>while</b>	<b>def</b>	<b>finally</b>	<b>in</b>	<b>print</b>
<b>continue</b>	<b>exec</b>	<b>import</b>	<b>pass</b>	
<b>class</b>	<b>except</b>	<b>if</b>	<b>or</b>	<b>yield</b>

Python 保留字



# 變數 ( Variable )

---

- Python 是物件導向的程式語言
- 物件包含三個要素

**ID**

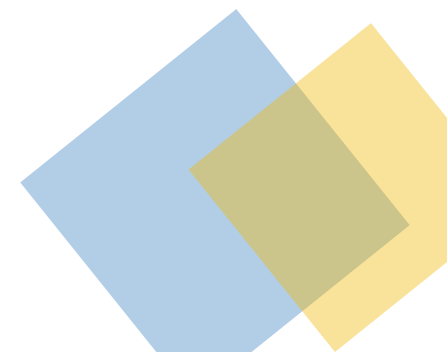
唯一識別

**Type**

物件型態

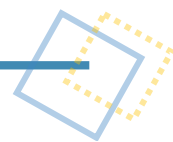
**Value**

物件的值





# 變數 ( Variable )



```
In [7]: a = 123
```

## ID

唯一識別

```
In [8]: id(a)
```

```
Out[8]: 140716787810960
```

## Value

物件的值

```
In [9]: type(a)
```

```
Out[9]: int
```

## Type

物件型態

```
In [10]: print(a)
```

```
123
```

# 資料型態

- 數值型態
- 字串型態
- 邏輯型態
- 容器型態

## #數值型態

Integers = 5

Float = 3.9 #小數

## #字串型態

String = 'hello'

a = '蘋果'

我 = '123'

## #邏輯型態：布林值，有True & False

Boolean = True

## #容器型態

Tuple = (1,2,3) #元組，不可變動

List = [1,2,3,4] #列表

Set = {1,2,3,4,5,6} #集合

Dictionary = {'key': 'value'} #字典，key&value是一組的

# print()

---

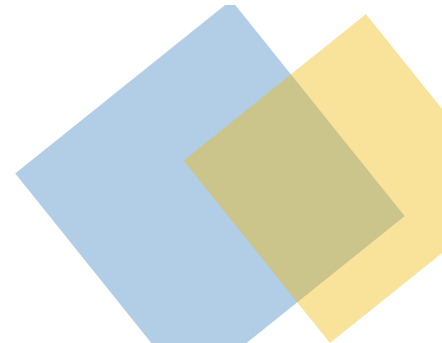
```
print(value, ..., sep=' ', end='\n')
```

```
print('a')  
print('b', end='.')  
print('c', end='/')
```

a  
b.c/

```
print('a', 'b', 'c')  
print('a', 'b', 'c', sep='/')
```

a b c  
a/b/c



# 型態轉換

- `int()`：轉為整數
- `float()`：轉為浮點數
- `str()`：轉為字串

```
t = 23.44555  
type(t)
```

`float`

```
q = int(t)  
q
```

`23`

```
type(q)
```

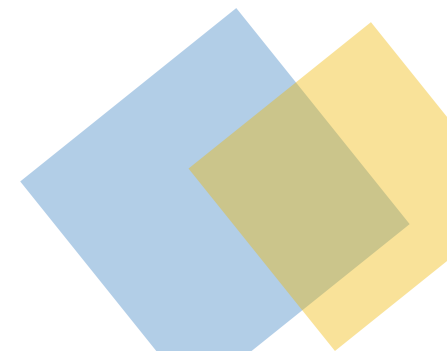
`int`

# eval()

---

```
eval(str)
```

- 計算字串中有效的表達式，並返回結果
- 將字串轉成相對應的型態
- 將被轉換為字串的變數，反轉回變數



# 數值型態

#數學運算

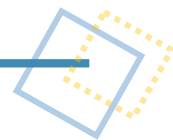
```
print(32 + 45) #加  
print(12 - 8) #減  
print(12 * 2) #乘  
print(12 ** 2) #指數
```

77

4

24

144



```
print(18 / 4) #除
```

```
print(18 // 4) #除取商
```

```
print(18 % 4) #除取餘
```

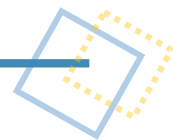
4.5

4

2

取商數時，在負值要小心!!

# 數值型態



- 大小比較：== 、 < 、 >

```
In [32]: 2 > 3
```

```
Out[32]: False
```

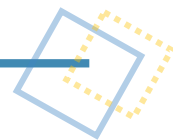
```
In [33]: 2 < 3
```

```
Out[33]: True
```

```
In [34]: 2 == 3
```

```
Out[34]: False
```

# 字串型態



- "" 或 '' 都可以

#字串相加

```
str1 = 'apple'  
str2 = 'pen'  
print(str1 + str2)
```

applepen

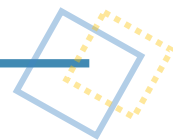
#重複字串

```
str1 = 'apple'  
print(str1*3)
```

appleappleapple



# 字串型態



```
num = 123
str3 = 'test'
print(num + str3)
```

-----

**TypeError**

Traceback (most recent call last)

<ipython-input-39-b77bf6c19808> in <module>

1 num = 123

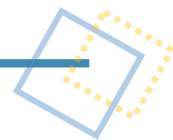
2 str3 = 'test'

----> 3 print(num + str3)

**TypeError:** unsupported operand type(s) for +: 'int' and 'str'

# 字串型態

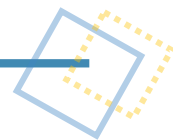
---



- 字串由多個字元組成

```
#字串長度 == 字元個數  
len(str1)
```

# 字串型態



- 找位置 ( index ) 相關的，皆使用 [ ]
- Index 從 0 開始，取頭不取尾

0      1      2      3      4  
a p p l e

```
str1 = 'apple'
```

```
str1[0]
```

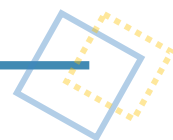
```
str1[-1]
```

```
str1[1:-2]
```

```
str1[:3]
```

# 字串型態

---



【 Notes 】 `str [ start : end : step ]`

- `start` : 字符的起始位置
- `end` : 字符的結尾位置
- `step` : 字符的間隔 ( 步長 )

```
str2 = 'hello, how are you?'
```

```
str2[1:9]
```

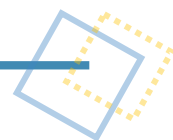
```
'ello, ho'
```

```
str2[1:9:2]
```

```
'el,h'
```

# 字串型態

---



【 Notes 】 `str.split(分割符, 最大分割次數)`

- 分割符不會出現在output上，他只是切分的依據

# 切割字串

```
str3 = 'hello, how are you?'  
str3.split('o')
```

```
['hell', ', h', 'w are y', 'u?']
```

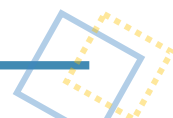
# 切割字串

```
str3 = 'hello, how are you?'  
str3.split('o', maxsplit=1)  
#maxsplit : 可以指定分割次數
```

```
['hell', ', how are you?']
```

# 字串型態

---



【 Notes 】 `str.replace(被替代, 替代, 替代次數)`

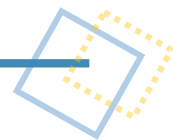
*# 取代*

```
str3 = 'hello, how are you?'
```

```
str3.replace('l', 'i') # 次數沒指定，預設全部取代
```

```
'heio, how are you?'
```

# 字串型態



- 字串格式化：把字串中的變數替換成變數值

```
# 字串格式化 (基本)
text = 'world'

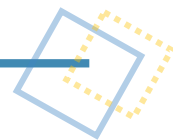
print('hello {}'.format(text))
```

hello world

```
a = 123
print(f"測試{a}")
```

測試123

# 邏輯型態



- 邏輯運算符：is、not、and、or

**AND    T    F**

<b>T</b>	<b>T</b>	<b>F</b>
----------	----------	----------

**F    F    F**

```
print(True and False)
```

False

**OR    T    F**

<b>T</b>	<b>T</b>	<b>T</b>
----------	----------	----------

**F    T    F**

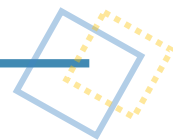
```
print(True or False)
```

True



# 容器型態 - tuple

---

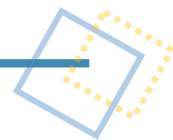


- tuple : 元組
- tuple 內的元素不可變動

```
1, 'Bob', 32
```

```
(1, 'Bob', 32)
```

# 容器型態 - tuple



```
test = (1, 'Bob', 32)
```

```
# 取值
```

```
test[2]
```

```
32
```

```
len(test)
```

```
3
```

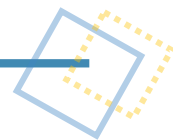
```
(1, 'Bob', 32) + (2, 'Bobb', 322)
```

```
(1, 'Bob', 32, 2, 'Bobb', 322)
```

```
(1, 'Bob', 32)*2
```

```
(1, 'Bob', 32, 1, 'Bob', 32)
```

# 容器型態 - list



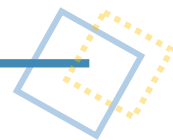
- list：列表、串列
- list 內的元素可以變動

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

```
a
```

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

# 容器型態 - list



- 在最後面增加元素

```
x = [1, 23, 34]
```

```
a.append(x)
```

```
a
```

```
[1, 2, 3, 4, 5, [1, 23, 34]]
```

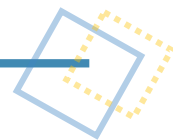
```
a.extend(x) #拆開
```

```
a
```

```
[1, 2, 3, 4, 5, 1, 23, 34]
```

# 容器型態 - list

---



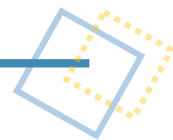
- 指定位置插入

```
a.insert(0, 23)
```

```
a
```

```
[23, 1, 2, 3, 4, 5]
```

# 容器型態 - list



- 提取：取出，並在 list 中將他刪除

```
list.pop(index)
```

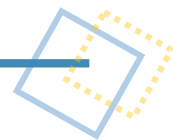
```
a.pop(0)
```

1

a

[2, 3, 4, 5]

# 容器型態 - list



- 刪除，遇到的第一筆

```
list.remove(要刪除的值)
```

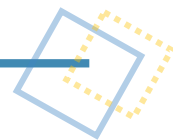
```
a.remove(2)
```

```
a
```

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

# 容器型態 - list

---



- 找出 x 的第一個 index ( 位置 )

```
list.index(要尋找的值)
```

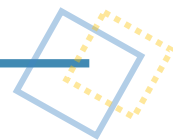
```
a.index(2)
```

1



# 容器型態 - list

---



- 找出 x 的第一個 index ( 位置 )

```
list.count(要計算次數的值)
```

```
a.count(1)
```

2

# 容器型態 - list



- 排序

`list.sort()` : 排序

`list.reverse()` : 反轉

```
b = [1, 75, 3, 65, 19]
```

```
b.sort()
```

```
b
```

```
[1, 3, 19, 65, 75]
```

```
b.reverse()
```

```
b
```

```
[75, 65, 19, 3, 1]
```

# 容器型態 - list

---

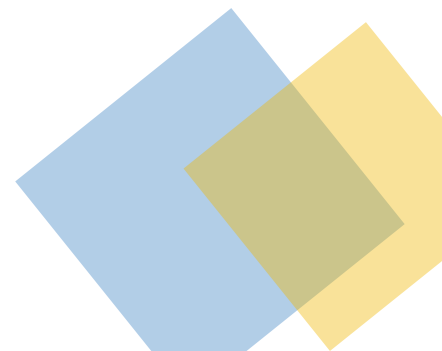
```
# 相加、相乘  
a = [1, 3, 3]  
b = [23, 4, 5]
```

```
a + b
```

```
[1, 3, 3, 23, 4, 5]
```

```
a*3
```

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



# 容器型態 - list

- 多維 list

```
#多維list
```

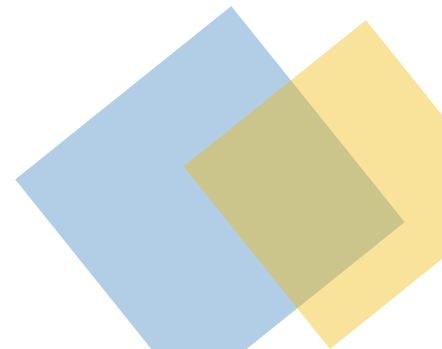
```
a = ['hey', '3', 'hi', [7, ['hello', 5, '9'], 9]]
```

```
len(a)
```

4

```
a[3][0]
```

7



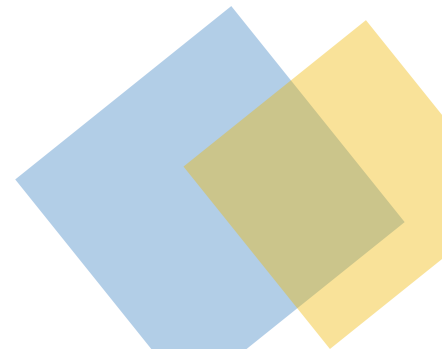
# 容器型態 - list

---

- 多維 list

```
int(a[1]) + a[3][0] + a[3][-1] + a[3][1][1] + int(a[3][1][-1])
```

33



# 容器型態 - dict

---

## 【Notes】

{ }：在查詢時，速度會比list快上很多

因為使用hash的技術，可以直接對應到我們要找的值

```
#宣告一個dict變數
```

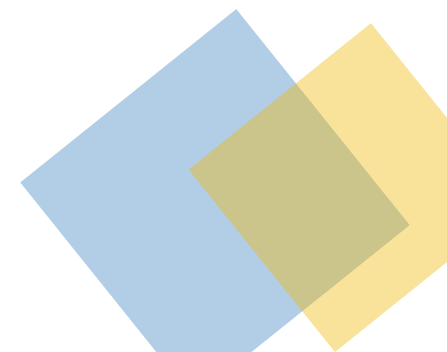
```
dict1 = {}
```

```
type(dict1)
```

dict

- dict：字典

映射類型：{'key':'value'}，一個key僅能對應到一個value



# 容器型態 - dict

```
dict1 = {'A':1, 'B':[1,2,3]}  
dict1
```

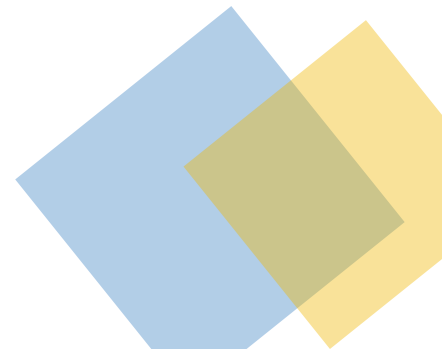
```
{'A': 1, 'B': [1, 2, 3]}
```

```
dict1['C'] = [5,6,7] #新增  
dict1
```

```
{'A': 1, 'B': [1, 2, 3], 'C': [5, 6, 7]}
```

```
dict1['B'] = [7,8,9] #更改  
dict1
```

```
{'A': 1, 'B': [7, 8, 9], 'C': [5, 6, 7]}
```



# 容器型態 - dict

- 查詢

# 所有的key (鍵)

```
dict1.keys()
```

```
dict_keys(['A', 'B', 'C'])
```

# 所有的values (值)

```
dict1.values()
```

```
dict_values([1, [7, 8, 9], [5, 6, 7]])
```

# 所有 key&values (鍵值對)

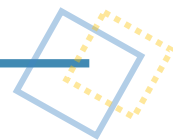
```
dict1.items()
```

```
dict_items([('A', 1), ('B', [7, 8, 9]), ('C', [5, 6, 7])])
```



# 容器型態 - dict

---



- 刪除：指定 key 刪除，其相對應的 value 也會隨之刪除

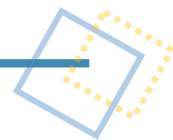
```
dict1.pop('A')
```

```
1
```

```
dict1
```

```
{'B': [7, 8, 9], 'C': [5, 6, 7]}
```

# 容器型態 - set



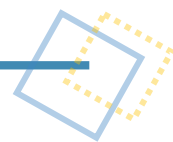
- 集合，無序、不重複的元素集
- 像是只有 key 的 dict

```
Set = set()  
type(Set)
```

```
SET = {1, 2, 3, 4}  
type(SET)
```

set

# 容器型態 - set



# List轉set

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

```
setA = set(a)
```

```
setA
```

```
{1, 2, 3, 4}
```

# 增加元素

```
SET.add(34)
```

```
SET
```

```
{1, 2, 3, 4, 34}
```

# 刪除元素

```
SET.remove(2)
```

```
SET
```

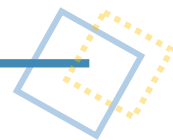
```
{1, 3, 4, 34}
```

# 判斷元素是否存在於集合中

```
1 not in SET
```

```
False
```

# 容器型態 - set



# 聯集

```
setA = {1, 2, 3, 4}
```

```
setB = {2, 3, 4}
```

```
setA | setB
```

```
{1, 2, 3, 4}
```

# 交集

```
setA & setB
```

```
{2, 3, 4}
```

# 插集 (注意! 順序)

```
setA - setB
```

```
{1}
```

```
setB - setA
```

```
set()
```



**THANKS**