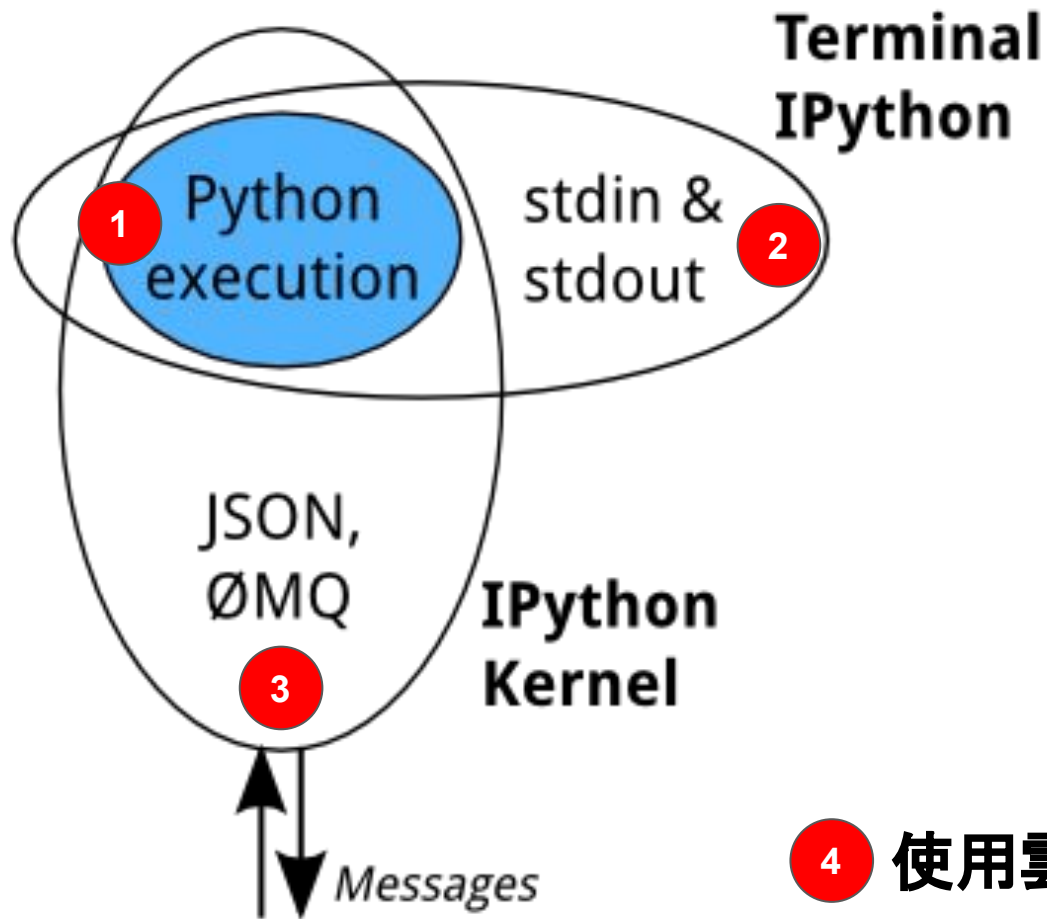


# Python基礎

## 講義網址：

- [https://github.com/vcdemy/python\\_basics](https://github.com/vcdemy/python_basics)
- <https://www.vcdemy.com>
- <https://khpy.teachable.com>

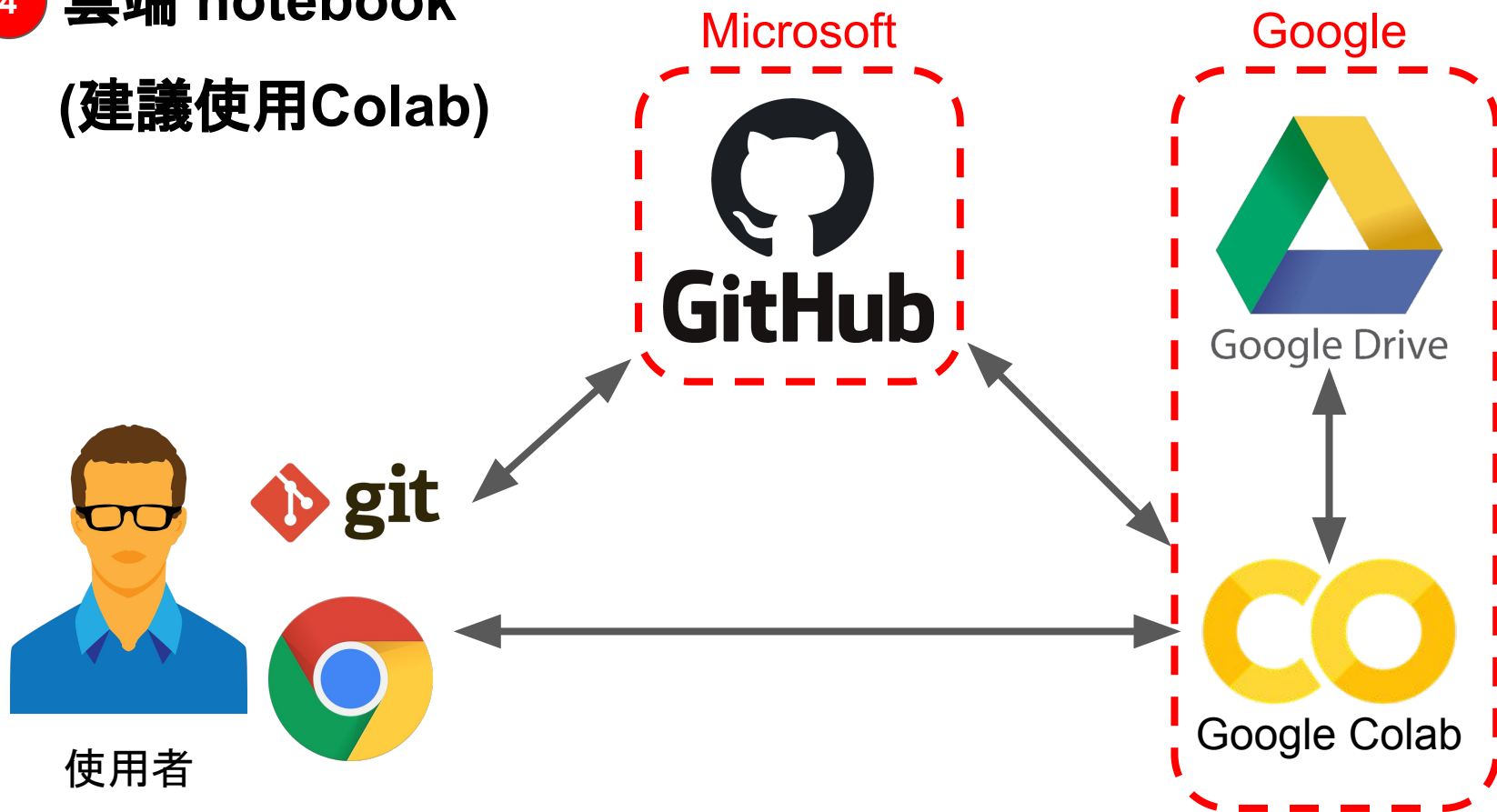
# Python的使用環境



建議初學者從這個開始！

**4** 使用雲端的 notebook

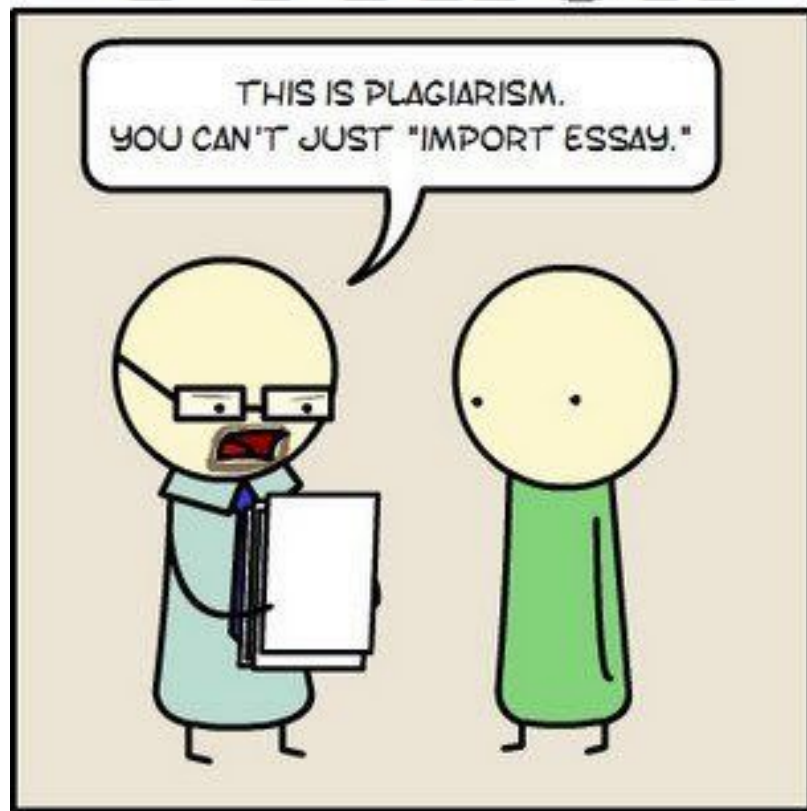
## 4 雲端 notebook (建議使用Colab)



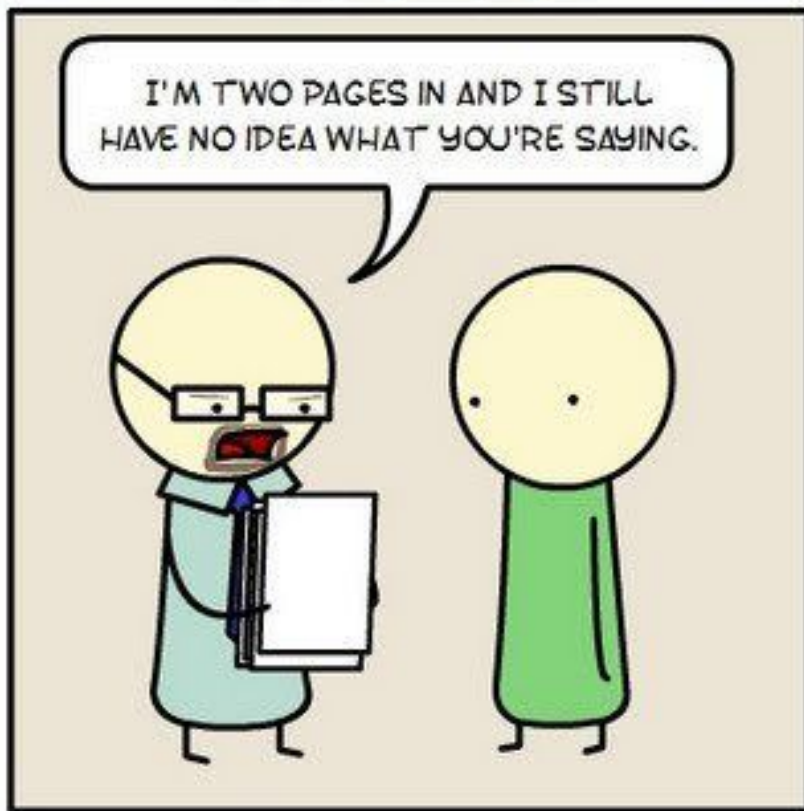
# DEMO

# Python的哲學

# PYTHON

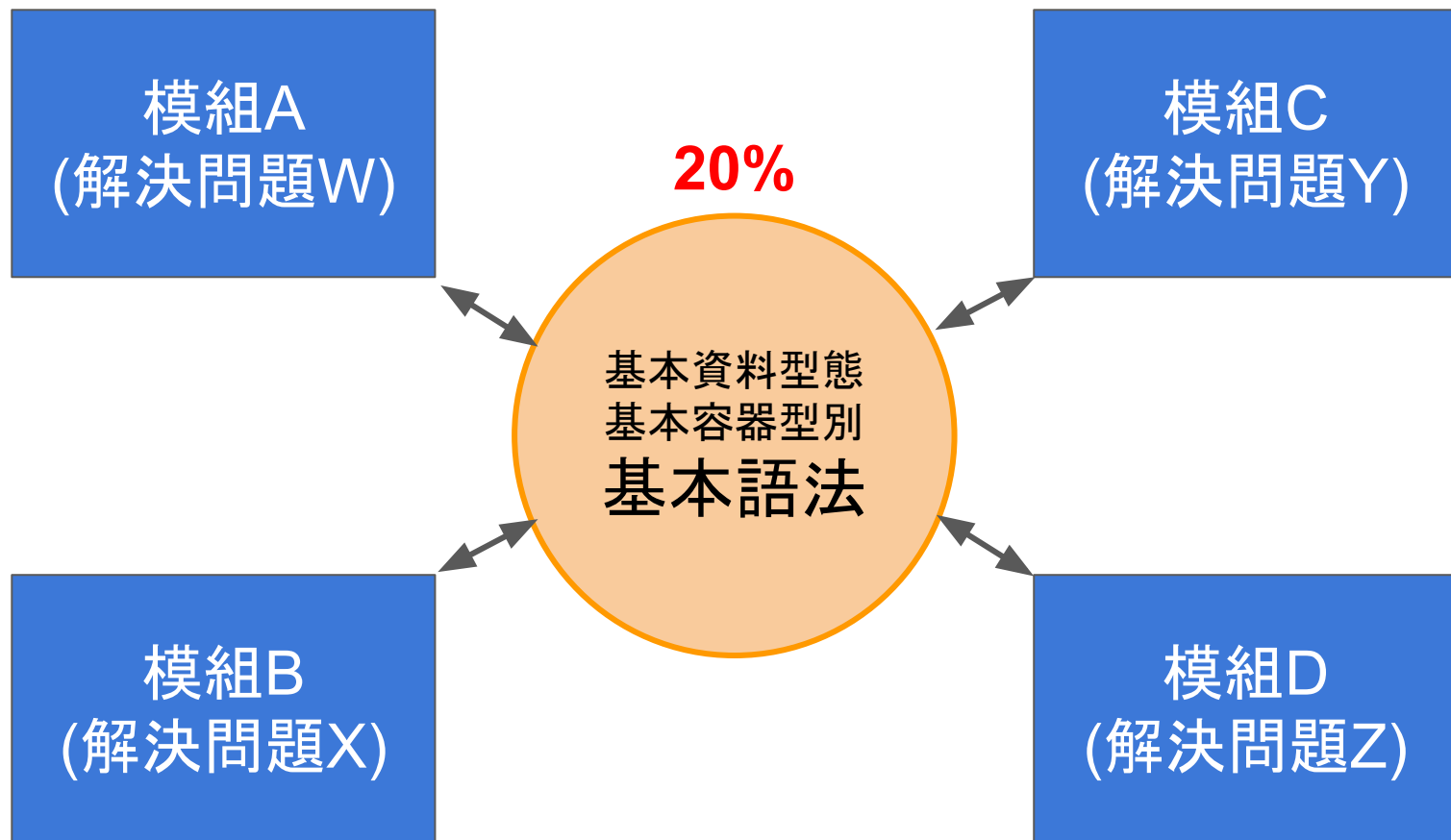


# JAVA









# 模組



.py檔

# 套件



管理.py檔的目錄結構

# Python 是膠水語言

- Python 有豐富的**模組及套件**可以使用。
- 使用者通常**不需要**實作程式所有的功能，而是需要協調及整合**套件及模組**的運作！！！！
- 使用 Python **可以在最短的時間做出產品雛形**。

# DEMO

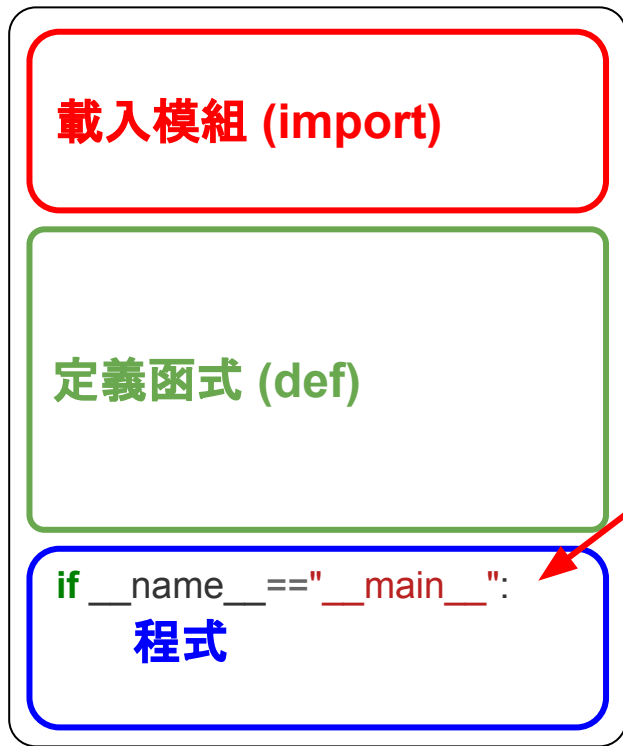
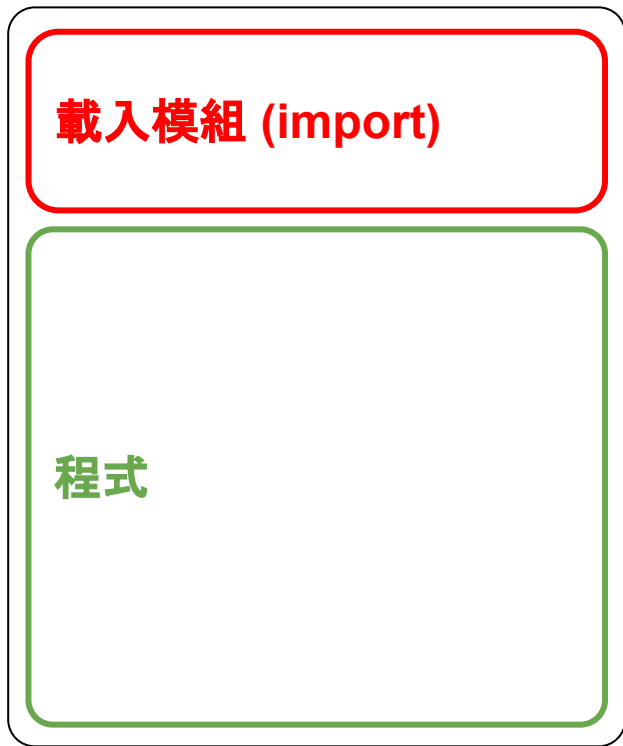
# Python 的檔案類型

# 主要兩種檔案類型

.py

.ipynb

# 常看到的 .py 檔的程式架構



判斷模組是直接被執行  
，還是被其他模組載入



# .ipynb 檔的程式架構

說明文字、數學式等等

Markdown Cell  
(Text Cell)

[ ]: 程式

輸出

Code Cell

## Jupyter notebook/lab 常用熱鍵及指令

- 執行cell:
  - Shift + Enter  $\Rightarrow$  執行該 cell, 並跳下一個 cell
  - Ctrl + Enter  $\Rightarrow$  執行該 cell, 但停在該 cell
  - Alt + Enter  $\Rightarrow$  執行該 cell, 並插入一個新的 cell
- 自動填滿: Tab 鍵
- 提示引數: Shift + Tab
- 求助:
  - ObjectName?
  - ObjectName??

# DEMO

# Python 的變數

什麼是變數？

存放資料的  
記憶體位址的別名

# 記憶體位址 (Memory Address)

沒有取別名記不起來



Memory Address	Value
0X7E0BECF6C0F0	0101 0101
0X7E0BECF6C0F1	0001 0101
0X7E0BECF6C0F2	0101 0001
0X7E0BECF6C0F3	1101 1101
0X7E0BECF6C0F4	1111 0001
0X7E0BECF6C0F5	0111 0111
0X7E0BECF6C0F6	0001 1101
0X7E0BECF6C0F7	0001 0100

# 變數的命名

- 變數名稱最好以**英文字母**或**底線**做開頭 (數字不能做開頭)
  - `_variable`, `variable`, `variable_`
- 除開頭字元外, 其他字元使用數字、底線、或英文字母
  - `_var123`, `var_123`
- 變數**會區分大小寫**
- 避開關鍵字 (keywords)
- 避開內建的物件 ([built-in objects](#))

# 寫程式！！！不是算數學！！！！

指派  
(assignment operator)

**X = X + 1**

變數記憶體位置  
(L-value)

變數內含的值  
(R-value)

可以使用 [Visualize Python](#) 來觀察變數跟記憶體的使用！



# 程式 = 資料結構 + 演算法

## 資料型態：

- **int**
- **float**
- **bool**
- **str**
- **None**

## 容器型別：

- **list** **[]**
- **tuple** **()**
- **dict** **{}**
- **set** **{}**

## 基本語法：

- **條件式(if)**
- **迴圈**
- **函式**
- **例外處理**

用 `type()` 檢視變數的資料型態

**`type(變數名稱)`**

# DEMO

# Python 的資料型態

什麼是**資料型態**？

資料型態告訴我們  
**變數裡面放了什麼**

## 四個基本資料型態 (變數裡面放了單一一個東西)

int



0

10

float



3.14

5.2

bool



True

False

str



“dog”

‘cat’

四個基本容器型別 (變數裡面放了可以放多個東西的容器)

[ ]

( )

{ }

{ }

list

tuple

dict

set

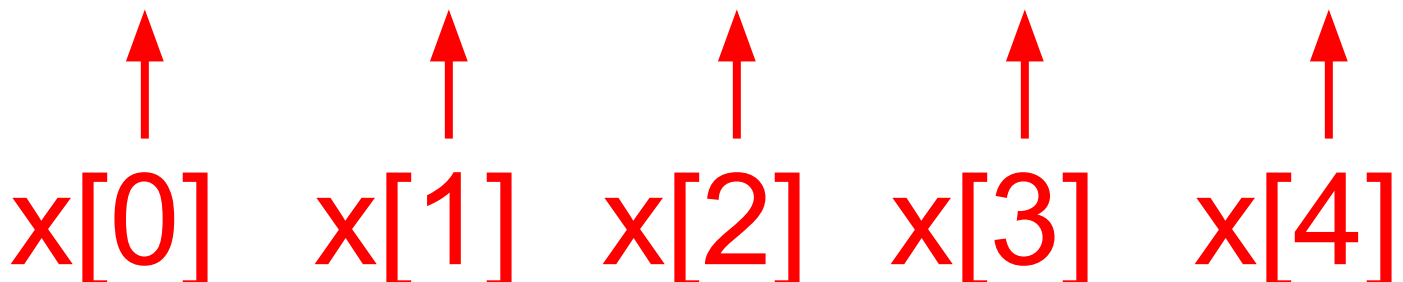
# List

```
x = ['a', 'b', 'c', 'd', 'e']
```



## Indexing

`x = ['a', 'b', 'c', 'd', 'e']`



The diagram illustrates array indexing for the list `x`. Red arrows point from the indices `x[0]` through `x[4]` to the corresponding elements in the list: `'a'`, `'b'`, `'c'`, `'d'`, and `'e'`. The opening and closing square brackets of the list are highlighted in red.

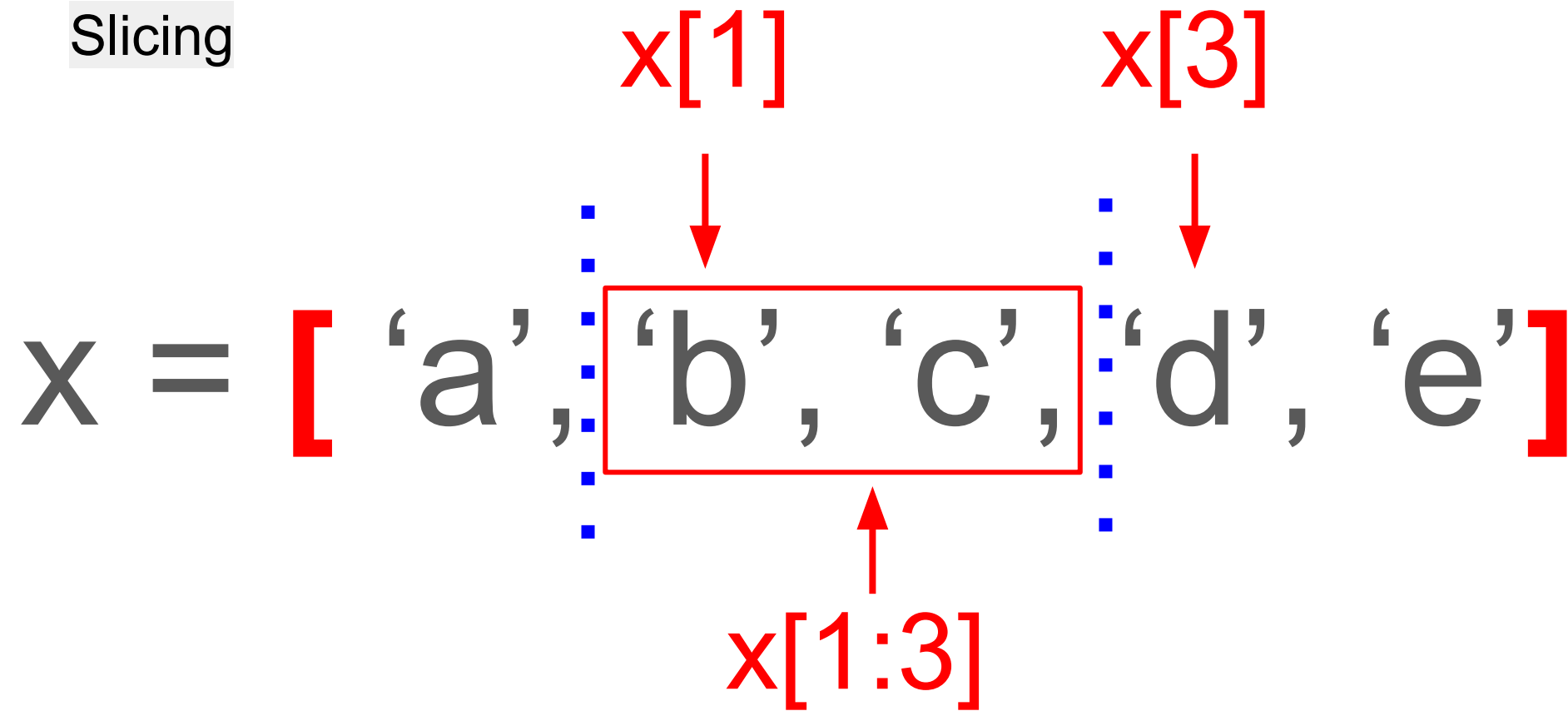
`x[0]`   `x[1]`   `x[2]`   `x[3]`   `x[4]`

## Indexing

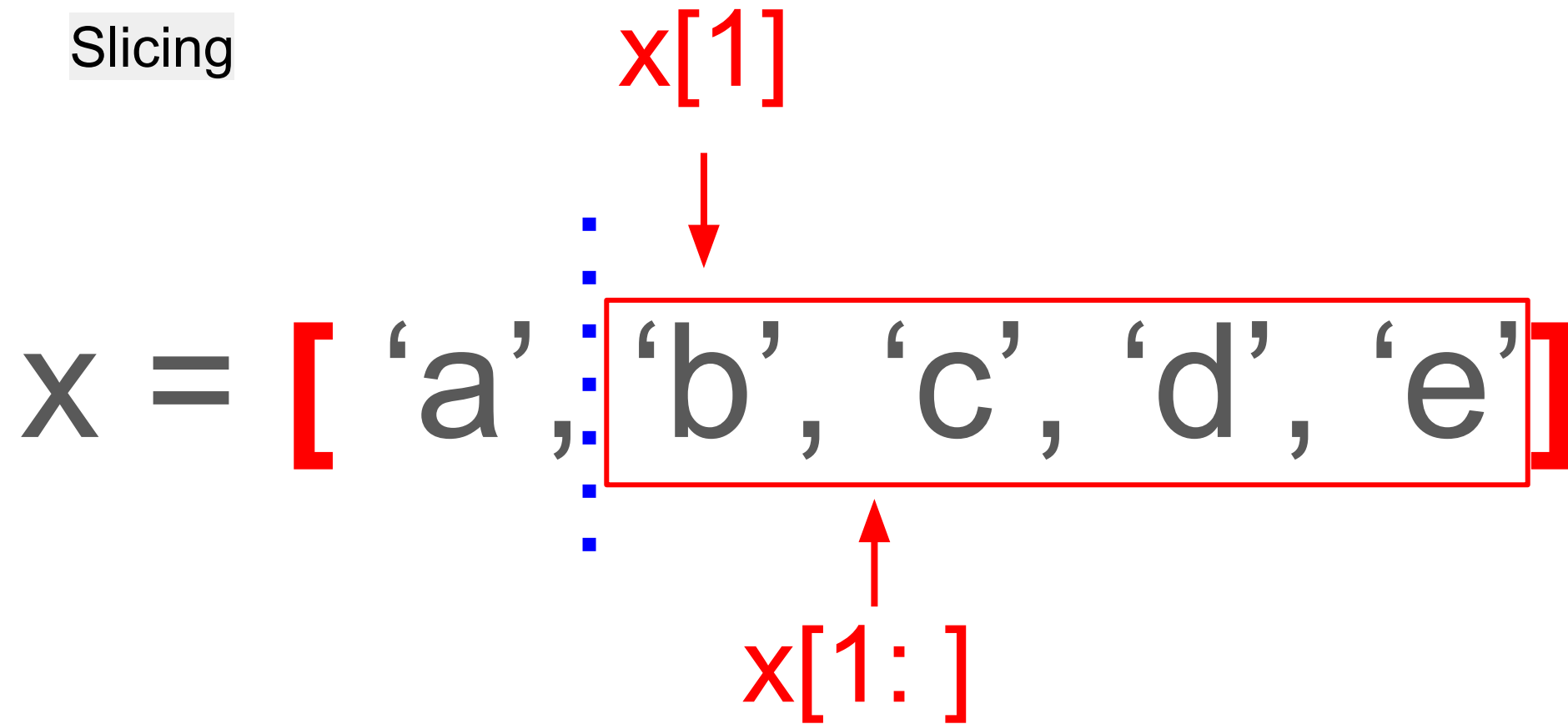
`x = ['a', 'b', 'c', 'd', 'e']`

`x[-5]` `x[-4]` `x[-3]` `x[-2]` `x[-1]`

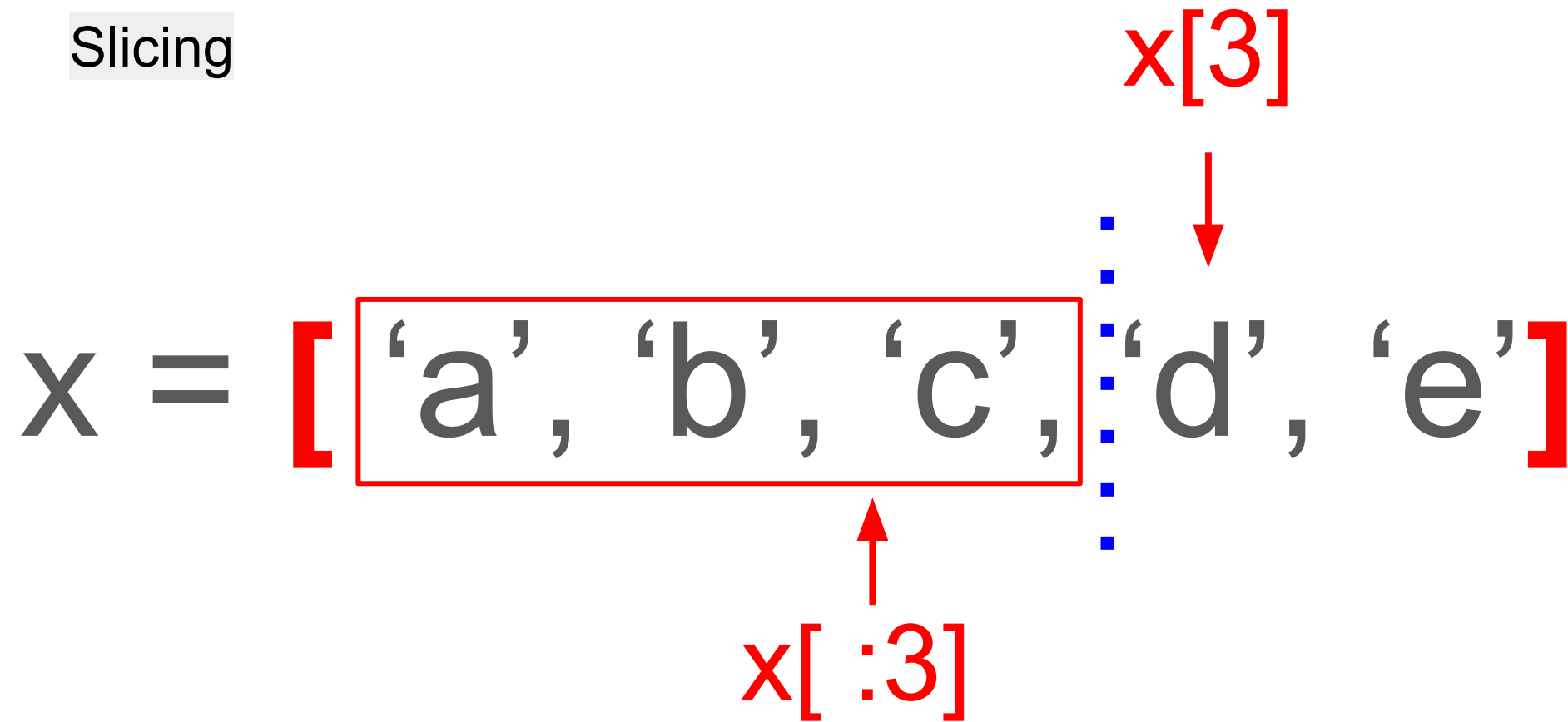
## Slicing



## Slicing



## Slicing



## Slicing

`x = [ 'a', 'b', 'c', 'd', 'e' ]`

The diagram illustrates list slicing on a Python list `x`. The list contains elements `'a', 'b', 'c', 'd', 'e'`. A red box highlights the slice `x[-3:]`, which includes elements `'c', 'd', 'e'`. A red arrow points from the label `x[-3]` to the start of the slice (index 2), and another red arrow points from the label `x[-3:]` to the same start index.

# Tuple

`x = ('a', 'b', 'c', 'd', 'e')`

## Indexing

$x = ('a', 'b', 'c', 'd', 'e')$

$x[0]$   $x[1]$   $x[2]$   $x[3]$   $x[4]$

The diagram illustrates array indexing for a tuple x. The tuple is shown as x = ('a', 'b', 'c', 'd', 'e') with red parentheses. Below each element, a red arrow points from a red index label to the element: x[0] points to 'a', x[1] points to 'b', x[2] points to 'c', x[3] points to 'd', and x[4] points to 'e'.



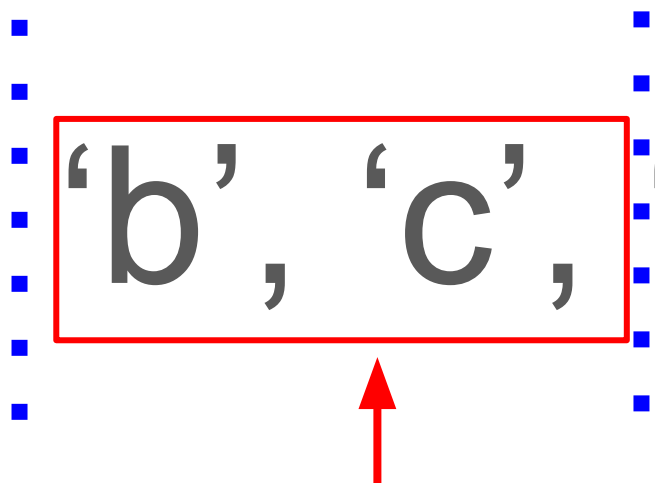
## Indexing

$x = ('a', 'b', 'c', 'd', 'e')$

$x[-5]$   $x[-4]$   $x[-3]$   $x[-2]$   $x[-1]$

## Slicing

`x = ('a', 'b', 'c', 'd', 'e')`



`x[1:3]`

# Dict

```
y = { 'apple': 10,  
      'orange': 20,  
      'banana': 30 }
```

**keys (鍵)**

**values (值)**

`y = { 'apple' : 10, 'orange' : 20, 'banana' : 30 }`

Diagram illustrating the assignment of values to a dictionary `y`:

- `y['apple']` points to the value `10`.
- `y['orange']` points to the value `20`.
- `y['banana']` points to the value `30`.

# Set

$x = \{'a', 'b', 'c', 'd', 'e'\}$

$y = \{'c', 'd', 'e', 'f', 'g'\}$

$x \& y = ?$        $x \mid y = ?$

# DEMO

# Python 的語法

**語法**可以做什麼？

可以讓程式**不是**  
**單調的從上到下執行**



# Python的語法重點

- Python 是用**縮排**來定義**程式區塊**。
  - 每一行程式前面的空白字元的數量是有意義的！
  - 每個程式區塊前的空白字元數量必須一樣。
- 使用 **import** 來**載入模組**
- **#** 開頭的文字為**註解**
- **引號**(單引號或雙引號)標註起來的為**字串**

## 四種基本語法

**條件式 (if statement):**

選擇性執行程式區塊。

**迴圈 (loops):**

重複執行程式區塊。

**函式 (functions):**

函式被呼叫時，執行函式內的程式區塊。

**例外處理(Exception Handling):**

嘗試執行程式區塊，發生錯誤時，執行例外處理程式區塊。

# 條件式 (if statements)

參考語法一：

```
if 條件 :  
    程式區塊
```

參考語法二：

```
if 條件 :  
    程式區塊1  
else:  
    程式區塊2
```

參考語法三：

```
if 條件1 :  
    程式區塊1  
elif 條件2 :  
    程式區塊2  
else:  
    程式區塊3
```

# 迴圈 (Loops)

for loop 參考語法：

```
for 變數 in 資料結構:  
    程式區塊
```

範例：

```
for i in range(5):  
    print(i)
```

while loop 參考語法：

```
while 條件:  
    程式區塊
```

範例：

```
i = 0  
while i < 5:  
    print(i)  
    i += 1
```

# 函式 (Functions)

函式定義參考語法：

```
def 函式名稱(引數):  
    程式區塊  
    return 回傳值
```

函式呼叫參考語法一：

```
函式名稱(引數)
```

函式呼叫參考語法二：

```
變數 = 函式名稱(引數)
```

# 例外處理 (Exception Handling)

例外處理參考語法：

```
try:  
    程式區塊  
except Exception as e:  
    例外處理程式區塊
```

# DEMO

**Thank You!**