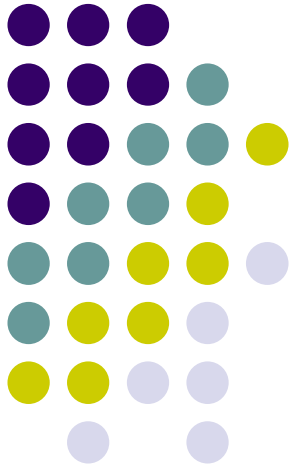
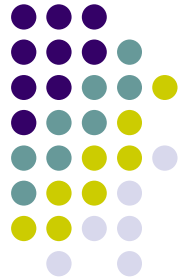
Python程式設計入門
基礎-複習

葉難

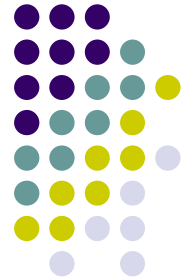


落差



- 課程是「Python程式語言」入門
- 內容必須盡量涵蓋Python語言所有功能
- 各位通常不想學「語言」，而是想要做事情
- 入門課程的範例都很簡單基礎，對已會某程式語言的人來說，很無聊
- 不情願的科學家

複習與提醒：語法

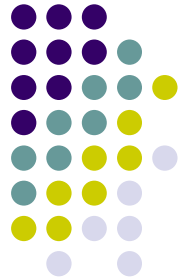


- 縮排不正確，出現異常IndentationError；執行Python直譯器時，加上參數「-t」可檢查是否混用空格與TAB
- 忘記「:」，冒號代表開始程式區塊，如if、for、def、else等，需縮排
- 程式區塊只有一條述句，可寫成一行但不建議

```
def foo(): print('hello')

if temp > 30: print('hot')
```

複習與提醒：程式碼太長



- 用「\ 加 換行字元」，延續到下一行

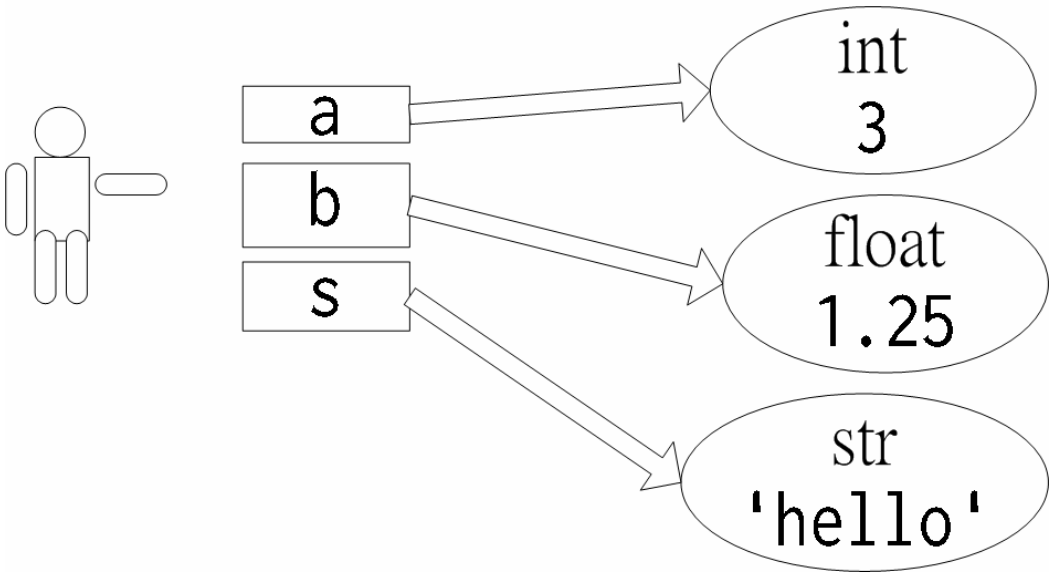
```
a, b, c, d, e, f = 1, 2, 3, 4 ,5, 6
if a > 0 and b > 1 and c > 2 and \
        d > 3 and e > 4 and f > 5:
    print('hello')
```

- 運用運算式的小括號「(、)」

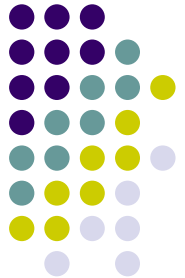
```
if (a > 0 and b > 1 and c > 2 and
        d > 3 and e > 4 and f > 5):
    print('hello')
```

複習與提醒：指派

- Python直譯器執行程式看到名稱時
- 若是第一次看到，會出現找不到的錯誤訊息，除非該名稱出現在指派述句「=」左邊
- 第一次的指派述句會產生名稱，並指向「=」右邊的物件



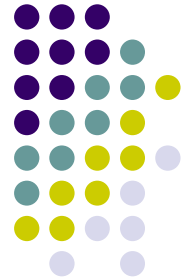
問題：先後順序



- 請問執行後，li的內容是什麼？

```
li = ['a', 'b', 'c', 'd']
i = 0
i, li[i] = 2, 99
```

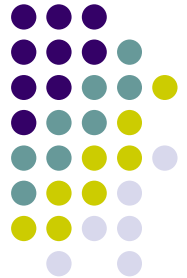
複習與提醒：物件的性質



- 別搞錯「=」與「==」，前者是指派，後者是比較相等與否
- 搞清楚可變和不可變；搞清楚執行動作的結果是建立新物件、還是原地修改（in-place change）

```
>>> a = b = 3
>>> a = 5
>>> li0 = li1 = [1, 2]
>>> li0 += [3, 4]          # [1, 2, 3, 4]
>>> li1 = li1 + [5, 6]  # [1, 2, 3, 4, 5, 6]
```

問題：既要取用、又要刪除



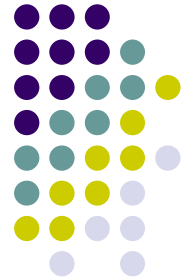
- 請問執行結果爲何？

```
li = [0, 1, 2, 3, 4, 5]

for i in range(len(li)):
    if li[i] % 2 == 0:
        del li[i]

print(li)
```

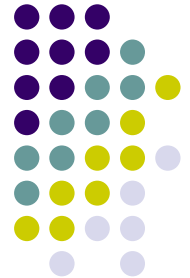
複習與提醒：動態型別



- 名稱與物件是兩樣東西，型別跟著物件
- Python直譯器執行動作（運算式）時，查知物件的型別，根據型別做動作
- 名稱就只是名稱，前一刻指向int物件，之後可改而指向list物件

- 動態：模糊地說是指「執行之時」
- 靜態：模糊地說是指「執行之前」

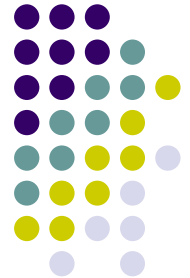
複習與提醒：型別



- 運算元型別不同，無法運算

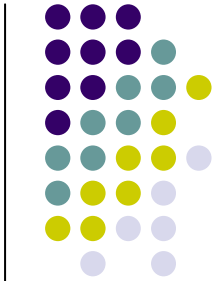
```
>>> 3 + 4.5                 # Python自動幫我們轉型
7.5
>>> 'abc' + 3
TypeError: Can't convert 'int' object to str implicitly
>>> 'abc' + str(3)     # 自己手動轉型
'abc3'
>>> int('123') + 456
579
```

複習與提醒：Python程式結構

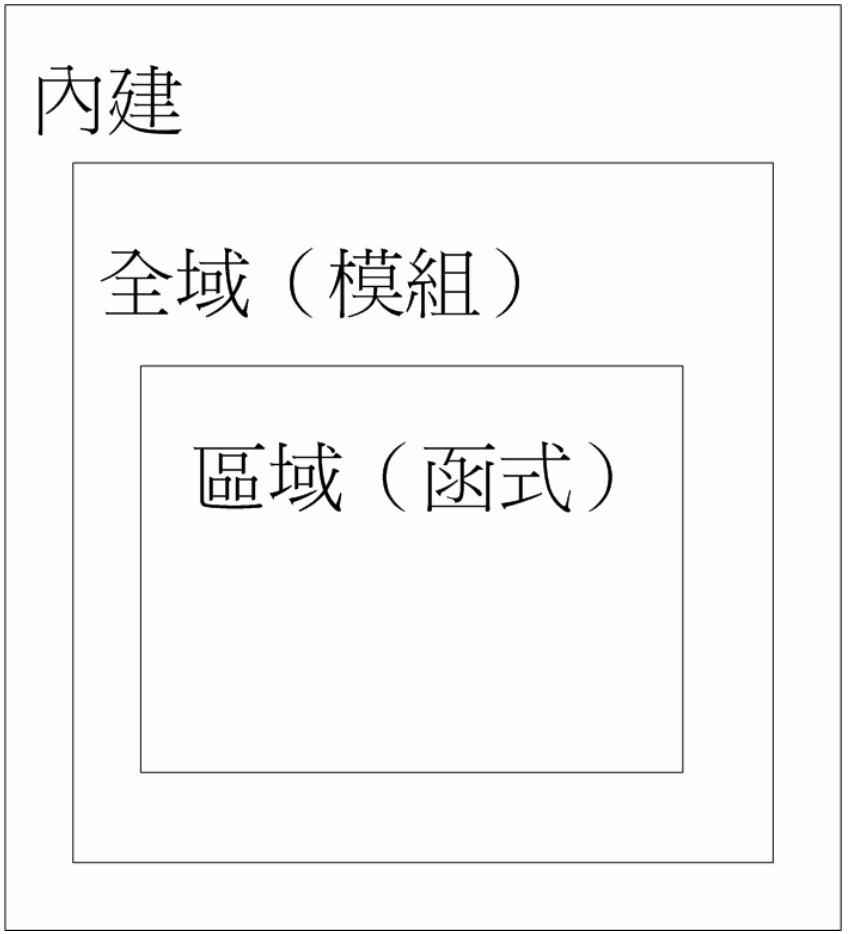


- Python程式由許多模組（module）組成，可能是Python程式檔案、也可能是C寫的
- 模組裡含有一堆述句（statement）
- 各述句皆有其語法和語意，含有運算式（expression）
- 運算式由運算子和運算元組成，建立與操作物件

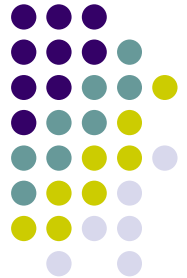
複習與提醒：範圍



- 依序尋找：區域（函式）、全域（模組）、內建
- 區域之外，其實還有個「外圍（閉包）」範圍
- LEGB：
  Local、Enclosing、Global、Builtin



匯入模組



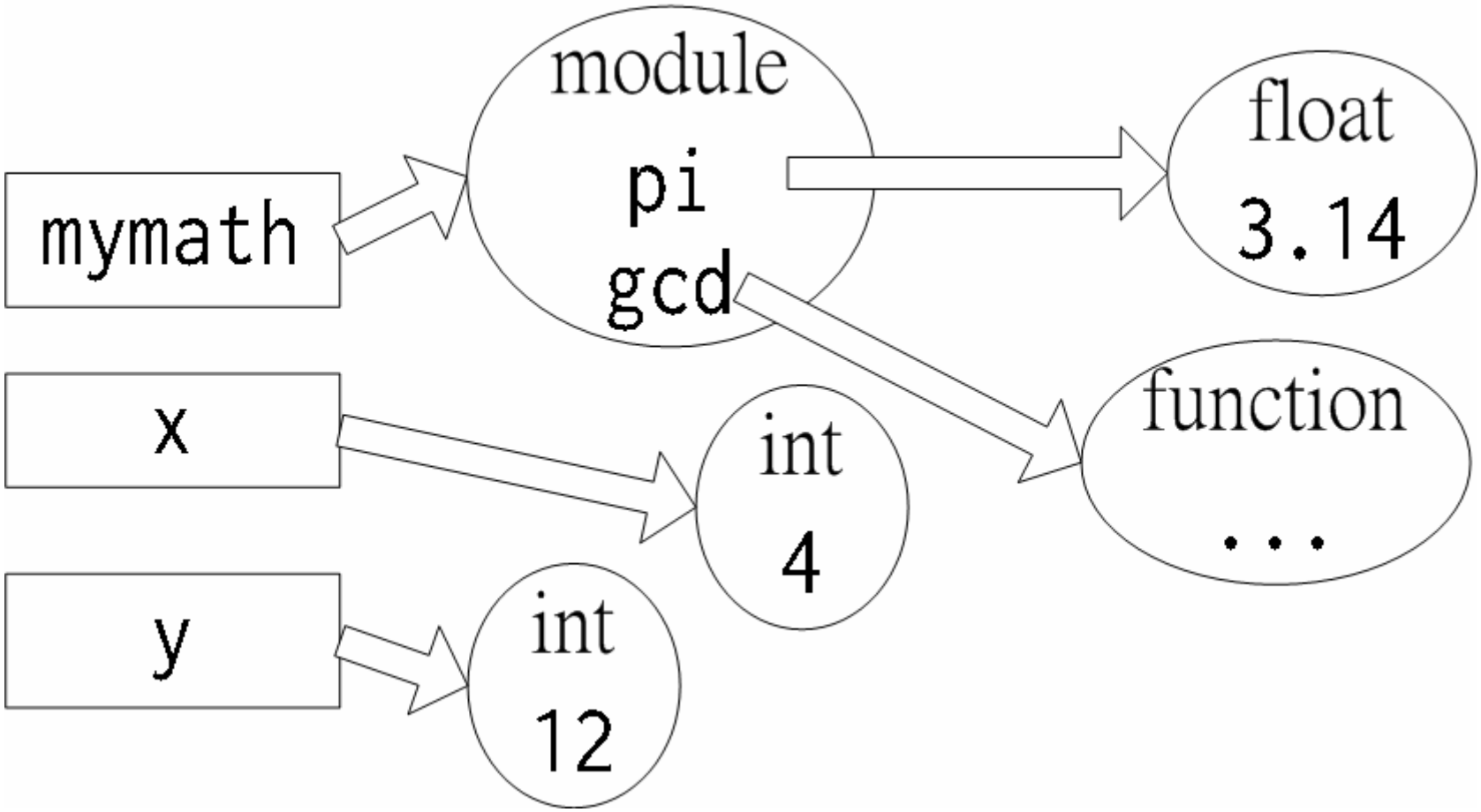
- 執行該模組，建立模組物件
- 產生名稱，指向物件

```
# hello.py
import mymath
print(mymath.pi)
x, y = 4, 12
print(mymath.gcd(x, y))
```

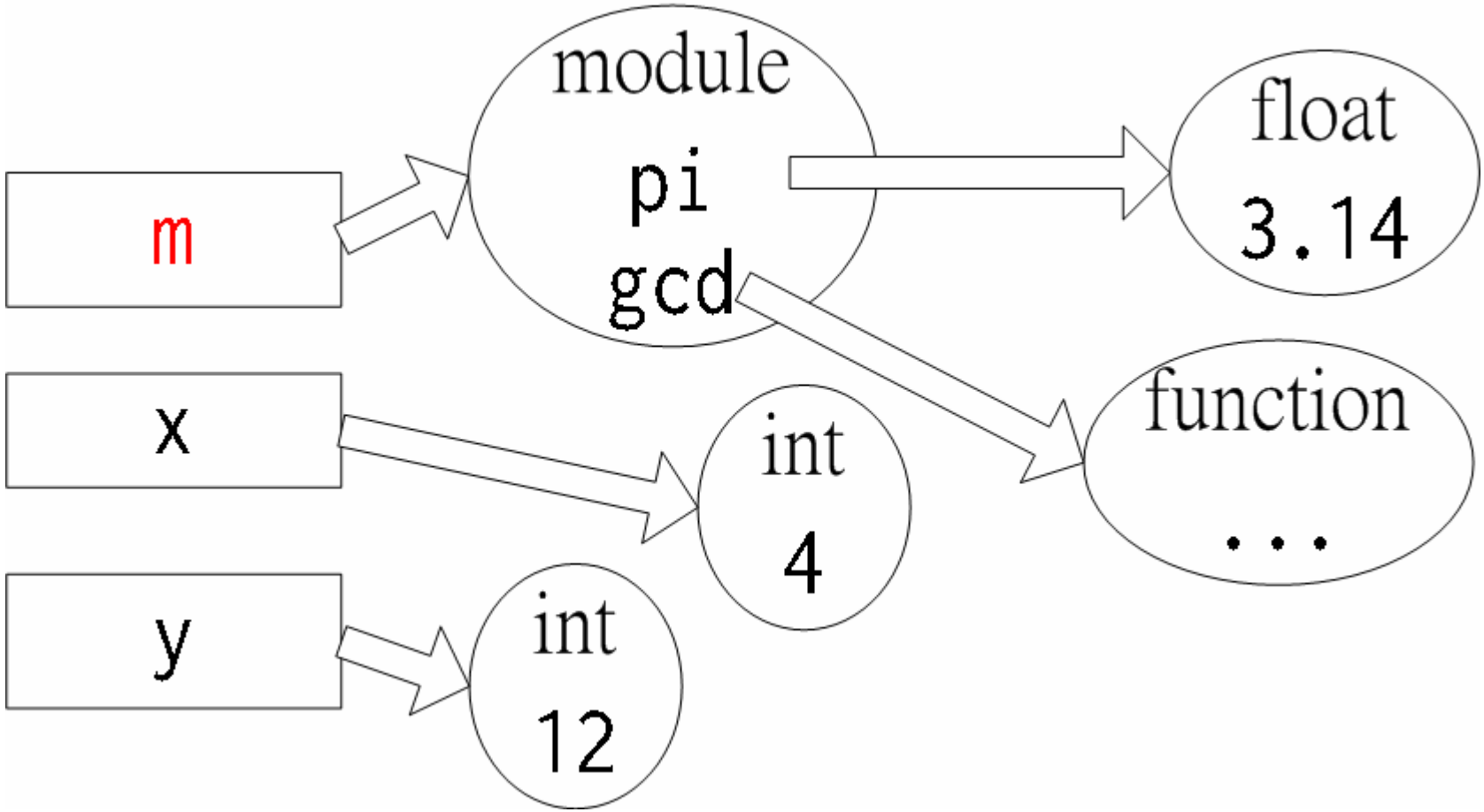
```
# mymath.py
pi = 3.14

def gcd(a, b)
    while b:
        a, b = b, a%b
    return a
```

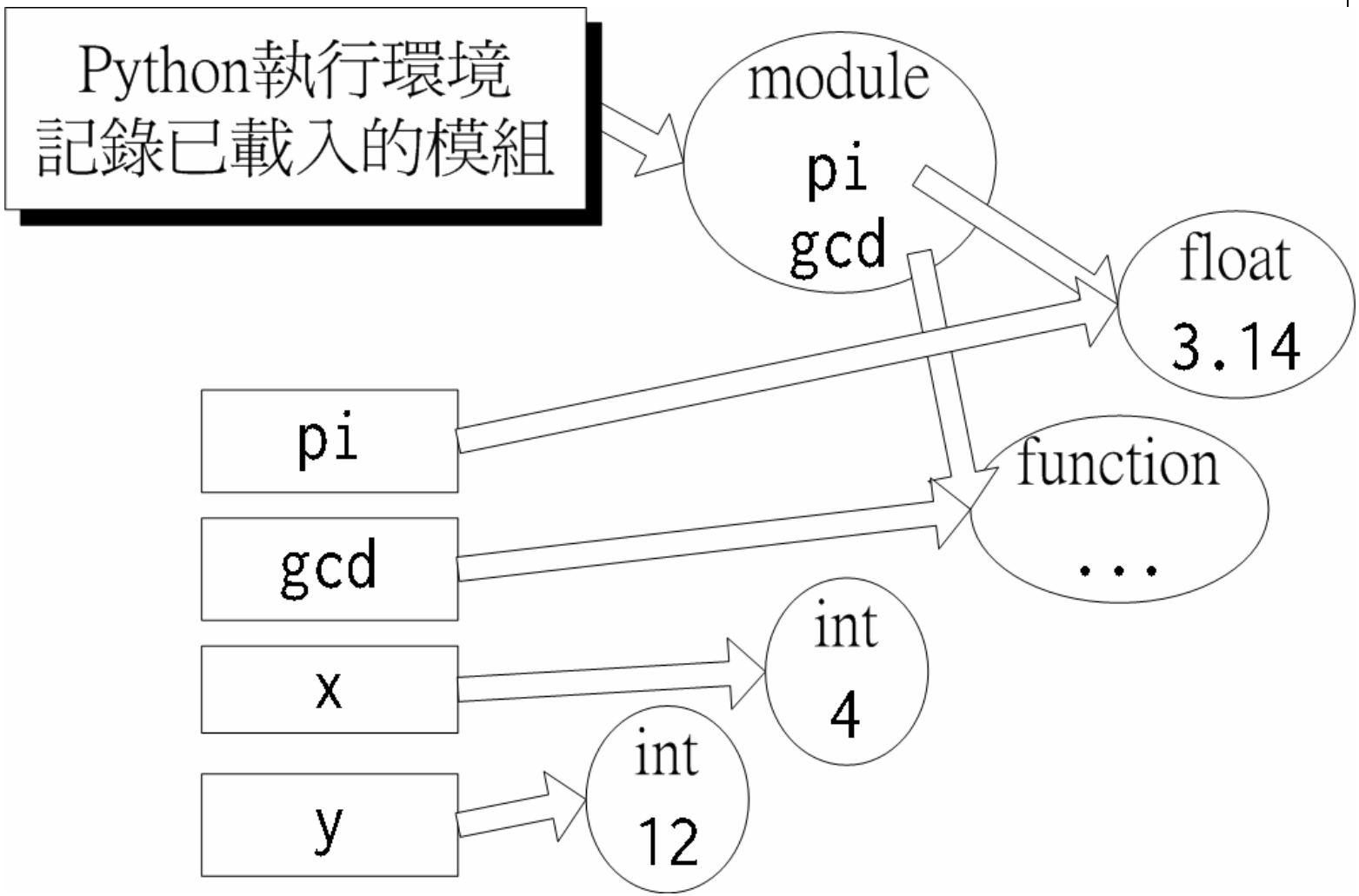
示意圖：**import mymath**



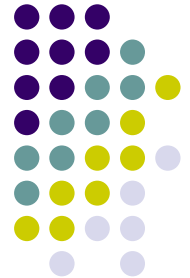
示意圖：import mymath as m



from mymath import pi
from mymath import gcd



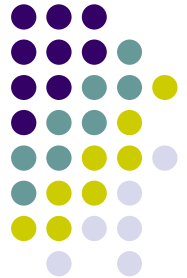
內建模組__builtins__



- 3.x版：builtins。2.x版：__builtin__

```
>>> __builtins__
<module 'builtins' (built-in)>
>>> __builtins__.len([0, 1, 2, 3, 4])
5
>>> dir(__builtins__)
['ArithmeticError', 'AssertionError', ...省
  略... 'tuple', 'type', 'vars', 'zip']
```

複習與提醒：**Python**官方文件



- https://docs.python.org/
- Library Reference：標準程式庫
- Library Reference
  2. Built-in Functions：內建函式
- Language Reference：語言規格書
- 注意版本：左上角落可選擇

Q&A