勞動部產業新尖兵計畫

人工智慧金融應用與實務培訓班

課程模組： AI 金融科技課程 - Python 程式設計

3. 控制敘述

葉建華 (Yeh, Jian-hua)

tdi.jhyeh@tdi.edu.tw
au4290@gmail.com

講次內容

- 選擇性 (Selection) 敘述
- 重複性 (Looping) 敘述
- 特殊的程式控制
- 函數 (Function)
- 變數的有效範圍
- 例外處理

程式的流程

- 程式有「循序性」：在結構化的程式設計中，程式的執行方式是循序的由第一行敘述依次往下執行
- 程式有「選擇性」：程式執行也可以依據條件的判斷而選擇執行某些敘述，並略過某些敘述的執行
- 程式有「重複性」：程式執行也可以依據特定的條件而重複的執行某些程式碼

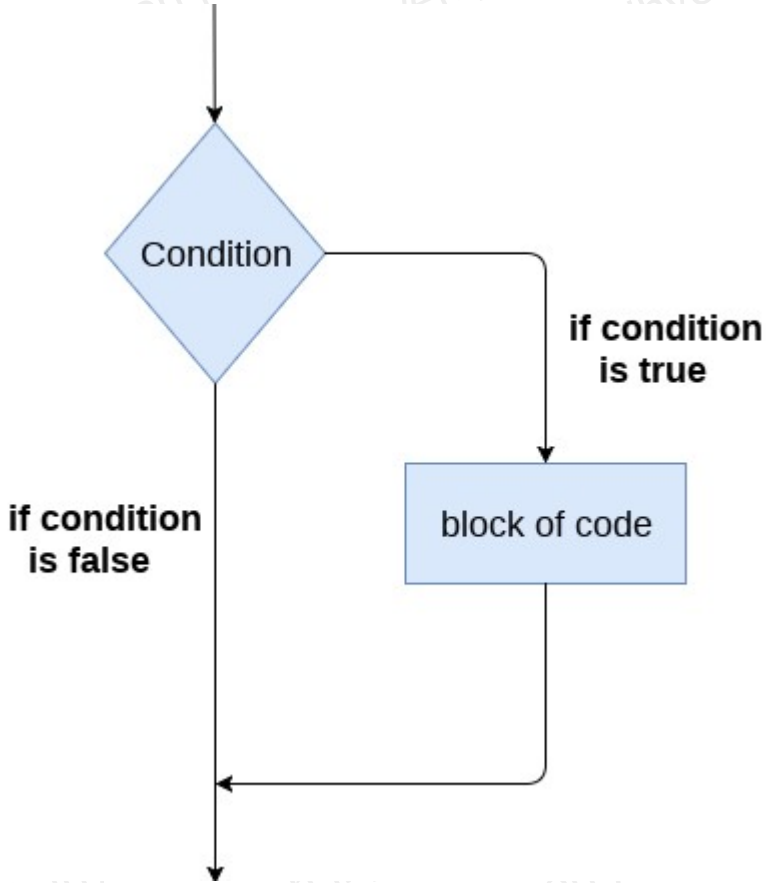
選擇性 (Selection) 敘述

- 選擇性敘述（或是稱為「條件判斷敘述」）的主要目的是讓程式可以依據不同的條件來執行不同區段的程式碼
  - 單項執行敘述 : 使用 if
  - 雙項執行敘述 : 使用 if-else
  - 多項執行敘述 : 使用 if-elif-else

單項執行敘述：if

- 如果條件式的值為 True，則執行指定的敘述，如果條件式的值為 False，則不執行

```
num = int(input("enter the number?"))
if num%2 == 0:
    print("Number is even")
```



程式要縮排（Indentation）？！

- Python 和許多程式使用括號 { } 來標示程式區塊不同， Python 使用縮排層次來標示
  - 一般而言會使用 4 個空白字元的縮排法

```
count = 0;
// strong typing
for (Student stu: students) {
    System.out.println(stu.getName());
    count++;
}
```

```
count = 0
# duck typing
for stu in students:
    print(stu.getName())
    count += 1
```

練習：找出三個數字中最大者

- 輸入三個數字，放入變數 a、 b、 c
- 使用 if 比較出最大的數字並印出該數字
- 怎麼做?

練習：找出三個數字中最大者

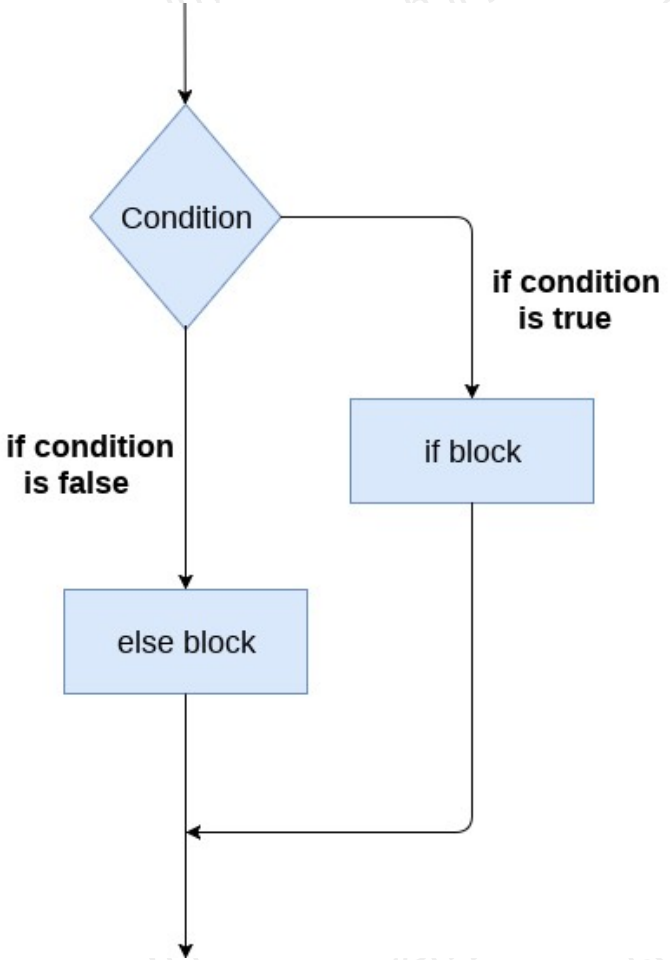
- 輸入三個數字，放入變數 a、 b、 c
- 使用 if 比較出最大的數字並印出該數字
- 怎麼做?

```
a = int(input("請輸入a:"))
b = int(input("請輸入b:"))
c = int(input("請輸入c:"))
if a>b and a>c:
    print("a最大")
if b>a and b>c:
    print("b最大")
if c>a and c>b:
    print("c最大")
```

雙向執行敘述：if-else

- 和單項執行敘述的功能相似，但增加了條件式不成立時，程式可以執行的區塊 else

```
age = int (input("Enter your age?"))
if age>=18:
    print("You are able to enter.")
else:
    print("Sorry, you have to wait.")
```



練習：回答數字是奇數還是偶數

- 輸入一個數字，放入變數 num
- 使用 2 的餘數計算，餘數為 0 則是偶數，否則是奇數
- 怎麼做?

練習：回答數字是奇數還是偶數

- 輸入一個數字，放入變數 num
- 使用 2 的餘數計算，餘數為 0 則是偶數，否則是奇數
- 怎麼做?

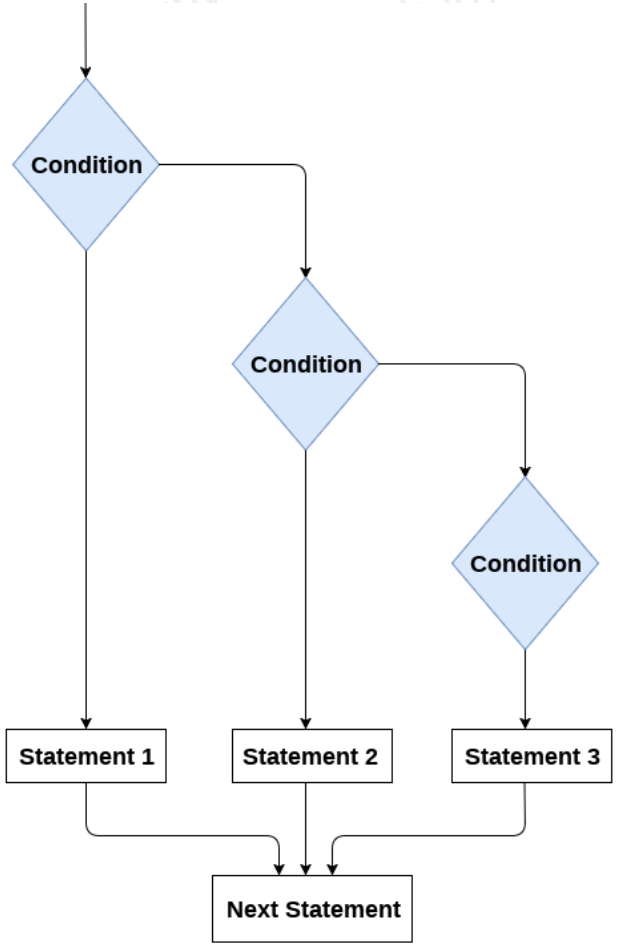
```
num = int(input("請輸入數字: "))
if num%2 == 0:
    print("此數為偶數")
else:
    print("此數是奇數")
```

是不是很簡單?

多向執行敘述：if-elif-else

- 對單項的條件式必須判斷多種不同的情況時使用

```
number = int(input("Enter the number?"))
if number==10:
    print("number is equals to 10")
elif number==50:
    print("number is equal to 50")
elif number==100:
    print("number is equal to 100")
else:
    print("number is not equal to 10, 50 or 100")
```



練習：從分數判斷成績的等級

- 輸入一個數字，放入變數 score
- 如果分數大於 85 分是 A 級， 60~85 分是 B+ 級， 40~60 分是 B 級， 30~40 分是 C 級， 30 分以下 ...
- 怎麼做?

練習：從分數判斷成績的等級

- 輸入一個數字，放入變數 score
- 如果分數大於 85 分是 A 級， 60~85 分是 B+ 級， 40~60 分是 B 級， 30 分以下 ...
- 怎麼做?

```
score = int(input("請輸入分數: "))
if score >= 85:
    print("恭喜! A級.")
elif score >= 60 and score < 85:
    print("B+級")
elif score >= 40 and score < 60:
    print("B級")
elif score >= 30 and score < 40:
    print("C級")
else:
    print("你怎麼會有這種分數? ")
```

巢狀的 if 敘述

- 如果程式中的 if 的指定敘述中又包含了其他的 if 敘述，則會形成巢狀 (Nested) 的 if 判斷區塊

```
x = int(input("Enter the number?"))
if x > 10:
    print("Above 10,")
    if x > 20:
        print("and also above 20!")
    else:
        print("but not above 20.")
```

三元運算子?

- 使用三元運算子 (ternary conditional operator)
  好處是程式碼可以看起來比較簡短

```
flag = True if x.isClick() else False

#等同於

if x.isClick():
    flag = True
else:
    flag = False
```

其實是想偷學英語子句結構 ...

```
# 短到爆
num = int(input('請輸入數字: '))
print('此數是偶數') if num%2==0 else print('此數是奇數')
```

pass?

- if 條件成立所要執行的敘述區塊不能為空
  - 萬一真沒有敘述要怎辦?
  - pass 吧!

```
a = 33
b = 200

if b > a:
  pass
```

講次內容

- 選擇性 (Selection) 敘述
- 重複性 (Looping) 敘述
- 特殊的程式控制
- 函數 (Function)
- 變數的有效範圍
- 例外處理

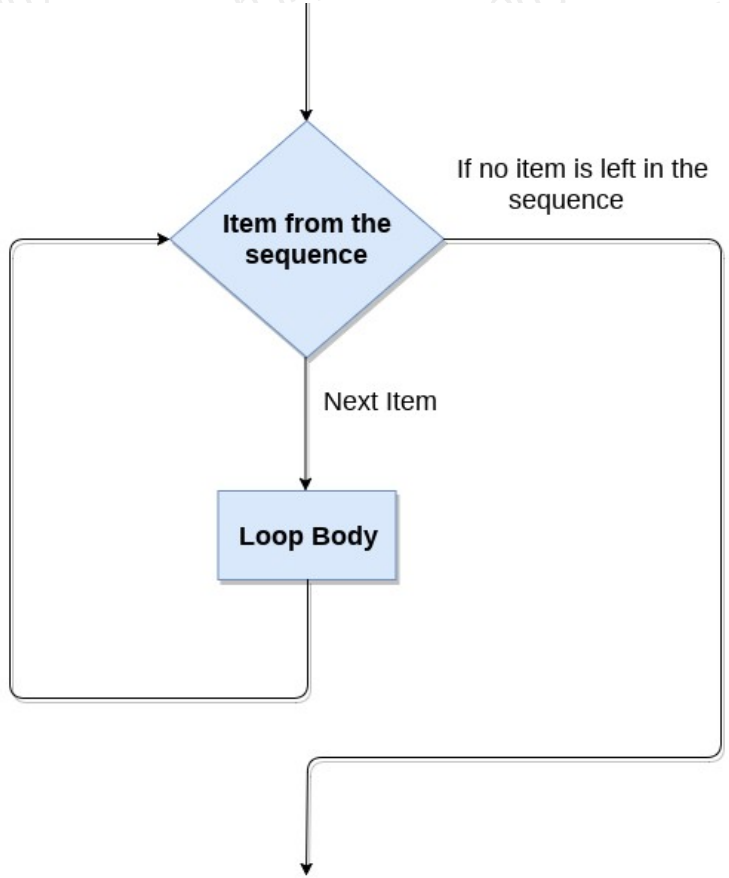
重複性敘述

- 主要目的是讓程式可以重複的執行某一區段中的程式碼：
  - for 迴圈
  - while 迴圈

for 迴圈

- 配合序列型別使用 (list, tuple, str)

```
str = "Python"
for i in str:
    print(i)

# 再來一個
list1 = [1,2,3,4,5,6,7,8,9,10]
n = 5
for i in list1:
    c = n*i
    print(c)
```



練習：把整個串列中的數字加總

- 建立一個串列 list1，資料內容是 [10, 15, 20, 25]
- 使用 for 迴圈將 list1 內的數字加總到 sum1 變數
- 印出 sum1 的值
- 怎麼做?

練習：把整個串列中的數字加總

- 建立一個串列 list1，資料內容是 [10, 15, 20, 25]
- 使用 for 迴圈將 list1 內的數字加總到 sum1 變數
- 印出 sum1 的值
- 怎麼做?

```
list1 = [10, 15, 20, 25]
sum1 = 0
for num in list1:
    sum1 += num
print(sum1)
```

使用 range 來配合 for

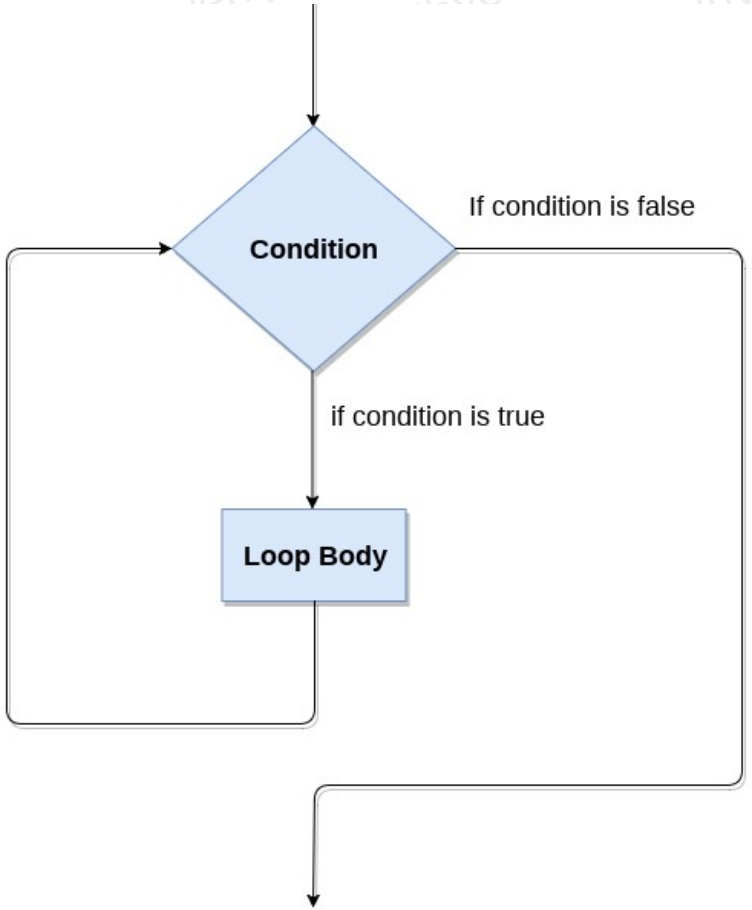
- 範圍型態 range： 表示不可變的數字序列
  - 常用來配合 for 迴圈

```
range1 = range(6)
for i in range1:
    print(i)

# 再來一個!
range2 = range(4, 7)
for i in range2:
    print(i)

# 再來一個!!
range3 = range(5, 10, 2)
for i in range3:
    print(i)
```

while 迴圈

- while 敘述多半是用於不可預測執行次數的情況下

```
num = int(input('請輸入數字: '))
print('倒數給你看! ')
while (num >= 0):
    print(num)
    num -= 1
print('數完啦! ')
```



練習：要使用者輸入特定數字

- 請使用者輸入一個數字，放入變數 num
- 如果 num 不等於 1234，則繼續問使用者
- 使用者輸入對了以後，印出「結束」
- 怎麼做?

練習：要使用者輸入特定數字

- 請使用者輸入一個數字，放入變數 num
- 如果 num 不等於 1234，則繼續問使用者
- 使用者輸入對了以後，印出「結束」
- 怎麼做?

```
num = 0
while num != 1234:
    num = int(input('請輸入數字: '))
print('結束')
```

else 可以配合 while 使用

- 當 while 條件不再成立時，可以執行 else 區段

```
i = 1
while i <= 5:
    print(i)
    i=i+1
else:
    print("The while loop exhausted")
```

巢狀迴圈敘述

- 如果將迴圈的語法合併使用，例如把 for 敘述中再包含另一個 for，甚至是 while 敘述，程式就形成了巢狀迴圈的架構了

講次內容

- 選擇性 (Selection) 敘述
- 重複性 (Looping) 敘述
- 特殊的程式控制
- 函數 (Function)
- 變數的有效範圍
- 例外處理

特殊的程式控制

- Python 提供了一些特殊的控制敘述，這些控制敘述可以進一步的控制迴圈的執行流程
  - break 敘述，用來跳脫迴圈區塊
  - continue 敘述，用來跳過本次迴圈的執行

break 敘述

- break 敘述可以用於終止迴圈的執行，並強迫跳離迴圈區段或跳離判斷式的執行區塊

```
list1 =[1,2,3,4]
index = 1;
for i in list1:
    if i == 4:
        print("item matched")
        index = index + 1
        break
print("found at", index, "location")
```

continue 敘述

- 會使程式略過 continue 敘述後的其他敘述的執行，並將程式執行權交還給迴圈控制敘述，讓迴圈控制敘述決定是否要繼續執行

```
i = 0
while i < 10:
    i = i+1
    if i == 5:
        continue
    print(i)
```

講次內容

- 選擇性 (Selection) 敘述
- 重複性 (Looping) 敘述
- 特殊的程式控制
- 函數 (Function)
- 變數的有效範圍
- 例外處理

函數的種類

- 函數：將特定功能的程式敘述組成一個區塊，並定義區塊名稱，而這個程式區塊可以被**重複**的使用
- Python 的函數粗分為兩大類
  - 使用者自訂的函數
  - Python 內建的函數

建立自訂的函數

- 使用 def 關鍵字

```
# 定義自訂的函數
def my_function():
    print("Hello from a function")

# 呼叫自訂的函數
my_function()
```

建立自訂的函數

- 可以透過函數的參數 (argument) 來傳遞訊息

```
# 定義自訂的函數
def my_function(name):
    print("Hello", name, '!')

# 呼叫自訂的函數
my_function('John')
my_function('Mary')
my_function('小智')
```

```
# 定義自訂的函數
def my_function(a, b):
    print("Sum of", a, "and", b, "is", (a+b))
    print("Product of", a, "and", b, "is", (a*b))
    print("Power of", a, "and", b, "is", (a**b))

# 呼叫自訂的函數
my_function(3, 2)
my_function(4, 3)
my_function(5, 3)
my_function('小智', 2)    # 這會發生什麼事?
```

建立自訂的函數

- 函數可以用 return 關鍵字來回傳函數運算結果

```
# 定義自訂的函數
def my_function(a, b):
    c = (a+b)*(a-b)
    return c

# 呼叫自訂的函數
x = my_function(3, 2)
print(x)
```

```
# 定義自訂的函數
def my_function(a, b):
    c = a+b
    d = a-b
    return c, d

# 呼叫自訂的函數
x, y = my_function(3, 2)
print("sum =", x)
print("diff =", y)
```

練習：計算長方形週長和面積

- 請使用者輸入兩個數字，分別放入變數 x, y
- x, y 分別是長方形的兩個邊
- 建立一個自訂函數，可以傳入長方形兩邊，計算並回傳長方形的週長和面積
- 怎麼做?

練習：計算長方形週長和面積

- 請使用者輸入兩個數字，分別放入變數 x, y
- x, y 分別是長方形的兩個邊
- 建立一個自訂函
  回傳長方形的週
- 怎麼做?

```
# 定義自訂的函數
def my_function(a, b):
    c = 2*(a+b)
    d = a*b
    return c, d

# 呼叫自訂的函數
x = int(input('請輸入長方形的其中一邊: '))
y = int(input('請輸入長方形的另一邊: '))
p, a = my_function(x, y)
print('長方形週長 = ', p)
print('長方形面積 = ', a)
```

傳參考呼叫?

- Python 中所有的函數都是傳參考呼叫
  - 意思是物件本身會傳遞進函數，操作這些物件可以造成物件內容變動
  - 但 Python 中有不可更改的物件，要記得

str, tuple, range 都是哦!

傳參考呼叫!

- 會變的

```
# 定義自訂的函數
def change_list(list1):
    list1.append(20)
    list1.append(30)
    print("list inside function = ",list1)

# 建立一個串列
list1 = [10, 30, 40, 50]
# 呼叫自訂的函數，傳入串列
change_list(list1)
# 串列內容變了!
print("list outside function = ",list1)
```

- 不會變的

```
# 定義自訂的函數
def change_string(str1):
    str1 = str1 + " Hows you "
    print("printing the string inside function :", str1)

# 建立一個字串
string1 = "Hi I am there"
# 呼叫自訂的函數，傳入字串
change_string(string1)
# 字串內容有變嗎? 沒有!
print("printing the string outside function :", string1)
```

練習：調整成績

- 建立一個成績串列 scores，資料內容是 [10, 40, 96, 77]
- 建立一個自訂函數，可以傳入成績串列 scores，每個分數都加上 20 分計算並回傳成績串列，但滿分最多 100
- 印出調整完的成績串列
- 怎麼做?

練習：調整成績

- 建立一個成績串列 scores，資料內容是 [10, 40, 96, 77]
- 建立一個自訂函數，可以傳入成績串列 scores，每個分數都加上
- 印出調整
- 怎麼做?

```
# 定義自訂的函數
def adjust_scores(scores):
    index = 0
    for score in scores:
        scores[index] = score+20 if score<80 else 100
        index += 1

scores = [10, 40, 96, 77]
adjust_scores(scores)
print(scores)
```

再談函數的參數

- 函數被呼叫的時候，使用的參數有以下幾種
  - 必要參數 (required argument)
  - 關鍵字參數 (keyword argument)
  - 預設參數 (default argument)
  - 不定長度參數 (variable-length argument)

必要參數

- 就是最常見的參數定義，只宣告參數名稱
  - 在呼叫函數時，各參數的值都必須提供

```
# 定義自訂的函數
def calculate(a, b):
    return a+b

calculate(10) # 錯! 有傳入資料給a，卻沒有給b
```

關鍵字參數

- 這是在函數呼叫的時候對參數「直呼其名」！

```
# 定義自訂的函數
def calculate(a, b):
    return a-b

calculate(10, 8)        # a=10, b=8
calculate(b=8, a=10)    # 硬是指定參數的值
```

預設參數

- 這是在函數定義時預先給參數預設值，函數被呼叫時如果外部有提供，則預設值失效

```
# 定義自訂的函數
def calculate(a, b=3):
    return a-b

calculate(10, 8)        # 答案是2
calculate(10)           # b使用預設值，答案是7
```

不定長度參數

- 在定義自訂函數時，參數名稱前加上 * 號即代表不定個數的參數

```
# 定義自訂的函數
def printme(*names):
    print("printing the passed arguments...")
    for name in names:
        print(name)

printme("John", "David", "Smith", "Nick")
```

使用遞迴

- 遞迴：函數的自我呼叫形式
  - 非常強大的邏輯表達方式
- 函數在呼叫時，系統會配置變數的使用空間
  - 所以遞迴在不斷的自我呼叫過程中 ...
  - 會耗用大量記憶空間

遞迴案例

- 階乘， n! = 1x2x3x…xn
  - 怎麼做?
- 費氏數列， f(n) = f(n-1)+f(n-2)
  - 怎麼做?
- 其他如輾轉相除法、河內塔、樹狀結構處理等等

使用遞迴解階乘問題

- 先定義遞迴關係
  - $f(n) = n \times f(n-1)$
- 定義遞迴關係最重要的一件事
  - 中止條件！ （你總得告訴程式什麼時候停）
  - 階乘問題的遞迴中止條件： n=1 時 f(1)=1

很重要！！

使用遞迴解階乘問題

- 先定義遞迴關係
  - f(n) = n x f(n-1)
- 定義遞迴關係最重要的一件事
  - 中止條件！（你總得告訴程式什麼時候停）
  - 階乘問題的遞迴中止條件： n=1 時 f(1)=1

很重要！！

```
def factorial(n):
    # 遞迴中止條件！！
    if n==1:
        return 1
    return n*factorial(n-1)

for i in range(10):
    n = i+1
    print(n, factorial(n))
```

```
1 1
2 2
3 6
4 24
5 120
6 720
7 5040
8 40320
9 362880
10 3628800
```

使用遞迴解費氏數列問題

- 先定義遞迴關係
  - f(n) = f(n-1)+f(n-2)
- 定義遞迴關係最重要的一件事
  - 中止條件！（你總得告訴程式什麼時候停）
  - 階乘問題的遞迴中止條件： f(0)=1, f(1)=1

使用遞迴解費氏數列問題

- 先定義遞迴關係
  - f(n) = f(n-1)+f(n-2)
- 定義遞迴關係最重要的一件事
  - 中止條件！（你總得告訴程式什麼時候停）
  - 階乘問題的遞迴中止條件： f(0)=1, f(1)=1

```
def fibonacci(n):
    # 遞迴中止條件！！
    if n==0 or n==1:
        return 1
    return fibonacci(n-1)+fibonacci(n-2)

# 遞迴中止條件！！
for i in range(10):
    n = i+1
    print(n, fibonacci(n))
```

```
1 1
2 2
3 3
4 5
5 8
6 13
7 21
8 34
9 55
10 89
```

講次內容

- 選擇性 (Selection) 敘述
- 重複性 (Looping) 敘述
- 特殊的程式控制
- 函數 (Function)
- 變數的有效範圍
- 例外處理

變數的有效範圍（Scope)

- 指在程式中可以使用某變數的地方
- 區域變數（Local Variable)：宣告在函數中，在執行區段中產生，離開區段後失效
- 內區域中可以使用外區段中的變數

變數的有效範圍（Scope)

- 指在程式中可以使用某變數的地方
- 區域變數（Local Variable)：宣告在函數中，在執行區段中產生，離開區段後失效
- 內區

```
def print_message():
    # message是區域變數
    message = "hello !! I am going to print a message."
    print(message)

print_message()
print(message)    # 錯！！message是區域變數啦!
```

講次內容

- 選擇性 (Selection) 敘述
- 重複性 (Looping) 敘述
- 特殊的程式控制
- 函數 (Function)
- 變數的有效範圍
- 例外處理

例外處理

- 例外是指程式流程中會發生不正常狀態，並導致執行中斷的狀況
  - 只要發生例外，程式會馬上中止執行
- Python 有許多內建的例外 ...

內建的例外

- ZeroDivisionError： 程式遇到了除以零的狀況
- NameError： 找不到名稱的狀況，可能是區域變數或是全域變數
- IndentationError： 程式縮排錯誤
- IOError： 任何輸出入動作所造成的錯誤
- EOFError： 檔案讀寫到了盡頭還在繼續操作…
- 其他還有很多

製造一下例外吧!

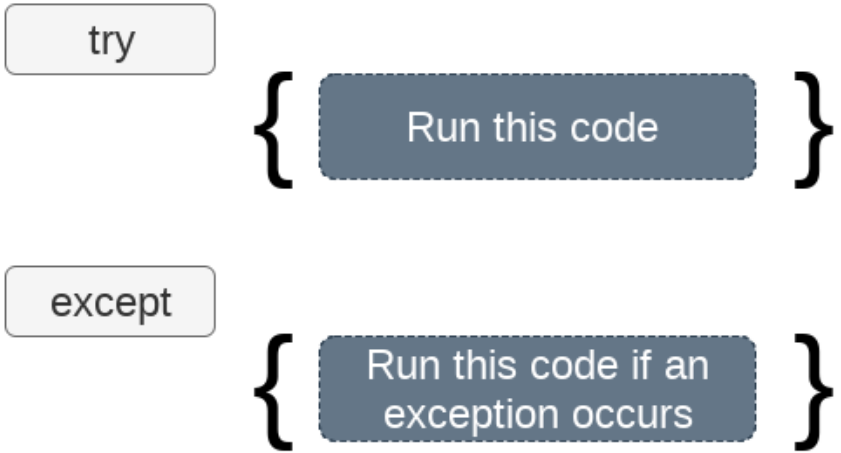
- 刻意產生一個 ZeroDivisionError

```
a = 75
b = 0
c = a/b
```

- 程式既然會中斷，那該怎麼辦?

例外處理框架

- try-except 敘述
  - try 區段中包含了可能產生例外的程式碼
  - except 區段表示需要處理的例外的程式碼



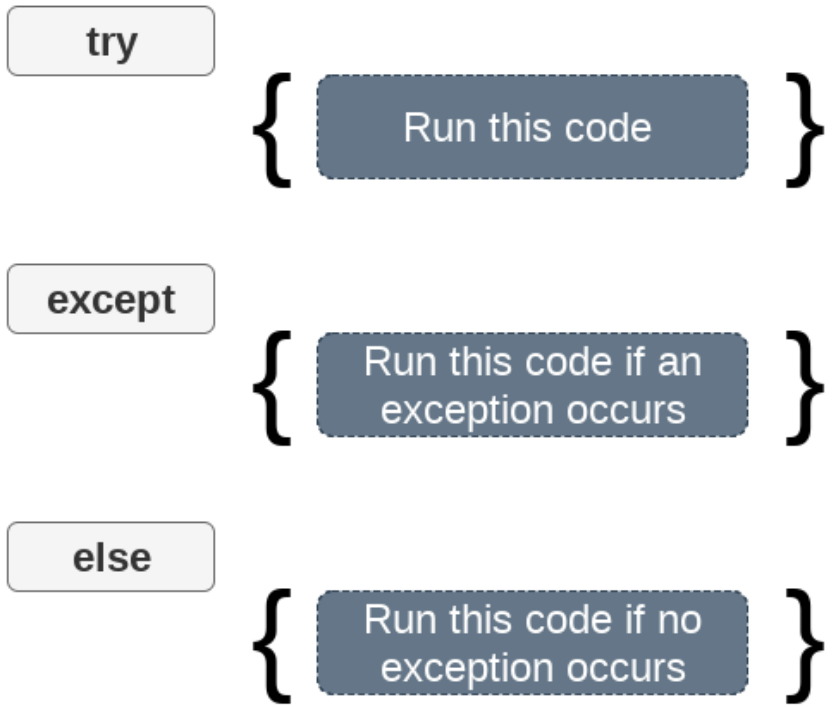
例外處理框架

- try-except 敘述
  - try 區段中包含了可能產生例外的程式碼
  - except 區段表示需要處理的例外的程式碼

```
try:
    a = int(input("Enter a:"))
    b = int(input("Enter b:"))
    c = a/b
except:
    print("Can't divide with zero")
# 這樣程式就可以繼續執行下去啦!
```

例外處理框架

- try-except-else 敘述
  - else 是用在當沒有例外發生時要執行的區段



使用例外物件

- 我們可以在 except 敘述將例外物件取出

```
try:
    a = int(input("Enter a:"))
    b = int(input("Enter b:"))
    c = a/b
    print("a/b = %d"%c)
# 取出例外物件，指定給變數e
except Exception as e:
    print("can't divide by zero")
    print(e)
else:
    print("Hi I am else block")
```

處理特定例外

- except 敘述可以指定處理特定例外

```
try:
    # 開啟檔案，如果檔案找不到，會發生例外
    fileptr = open("file.txt","r")
# 處理輸出入例外，檔案找不到例外是其中一種
except IOError:
    print("File not found")
else:
    print("The file opened successfully")
    fileptr.close()
```

處理特定例外

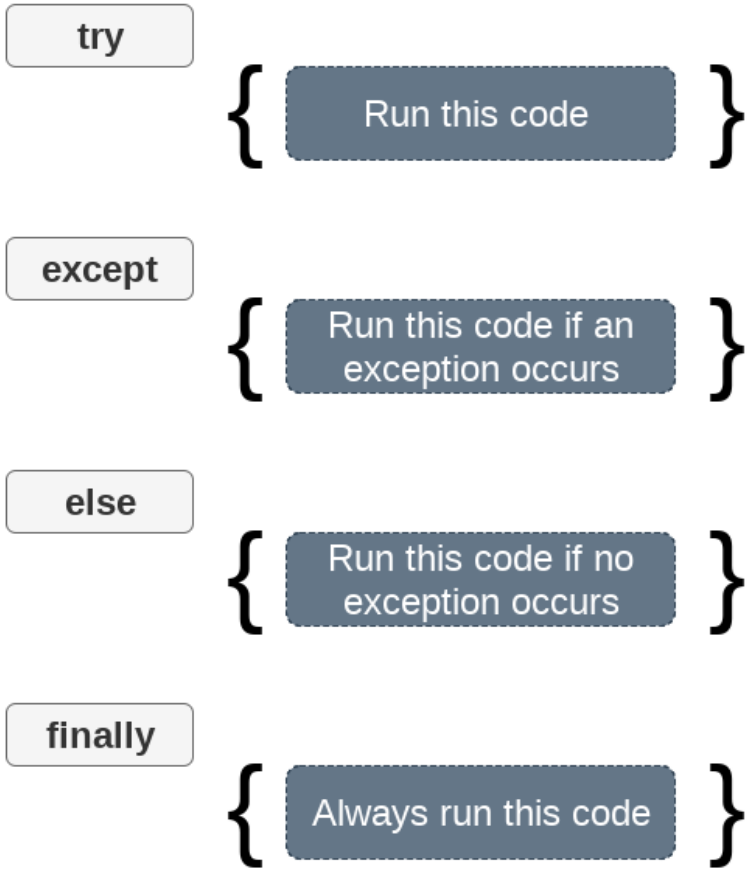
- 單一 except 敘述可以處理多個例外，也可以在單一 try 中用多個 except

```
# 平行指定
a,b = 1,0

try:
    print(a/b)
    print("This won't be printed")
    print('10'+10)
# 一次處理多種例外
except (TypeError, ZeroDivisionError):
    print("You got exception")
```

```
# 平行指定
a,b = 1,0

try:
    print(a/b)
    print("This won't be printed")
    print('10'+10)
# 處理第一種例外
except TypeError:
    print("You added values of incompatible types")
# 處理第二種例外
except ZeroDivisionError:
    print("You divided by 0")
```

Finally 敘述

- finally 區段是一個不論例外是否發生，它都一定會執行的一個區段



引發例外： raise

- 引發例外通常是要通知外部錯誤狀況
  - 常用在自訂函數中
  - 語法： raise 例外物件

```
try:
    num = int(input("Enter a positive integer: "))
    if num <= 0:
        # we can pass the message in the raise statement
        raise ValueError("That is a negative number!")
except ValueError as e:
    print(e)
```

自訂例外

- 例外可以自訂，但是目前不談

這個講次中，你應該學到了 ...

- 選擇性敘述如何撰寫
- 重複性敘述如何撰寫
- 函數如何撰寫，參數如何傳遞
- 例外處理的框架