一步一腳印
精通Python程式設計

程序及物件導向程式設計

課程講師: 陳建銘

2018/06/06

課程大綱

- 程序導向程式設計
  - 流程
  - 函式
  - Python內建函式
  - 呼叫自訂函式
- 物件導向程式設計
  - 類別
  - 繼承
  - 多型

程序導向程式設計

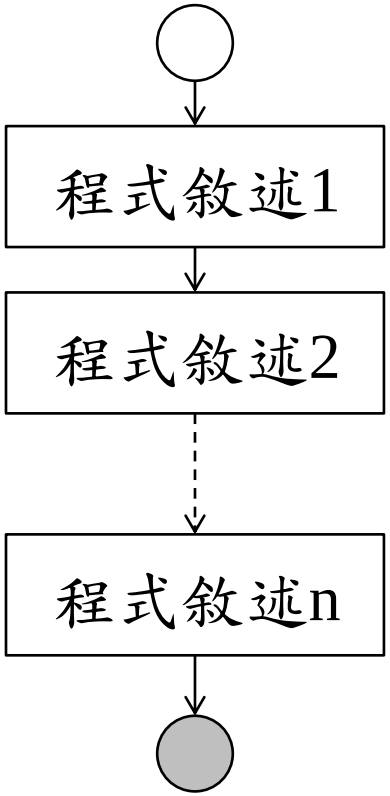
流程－

程式執行的方向稱為流程，流程可透過條件控制或迴圈來改變方向。

開發大型程式專案時，重複的動作會產生重複的程式碼，因此可透過函式將重複執行的功能包裝成功能模組。

模組化的目的是為了能夠重複使用被定義的功能。

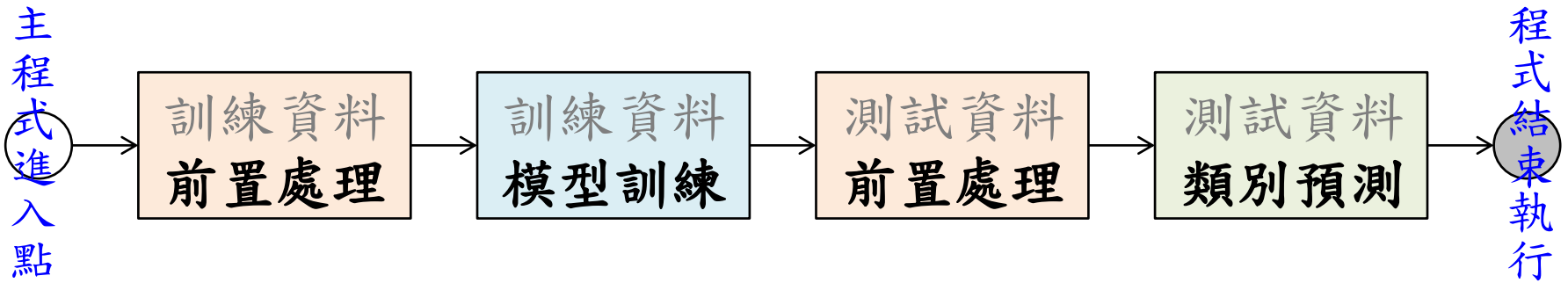
由於功能已被分割且模組化，因此在主程式中的程式敘述只是依序呼叫各函式來完成被指定的任務。



程式敘述除了包含宣告、控制、迴圈等內建語法及函式外，也包含了程式設計師自行設計的功能模組

程序導向程式設計 cont.

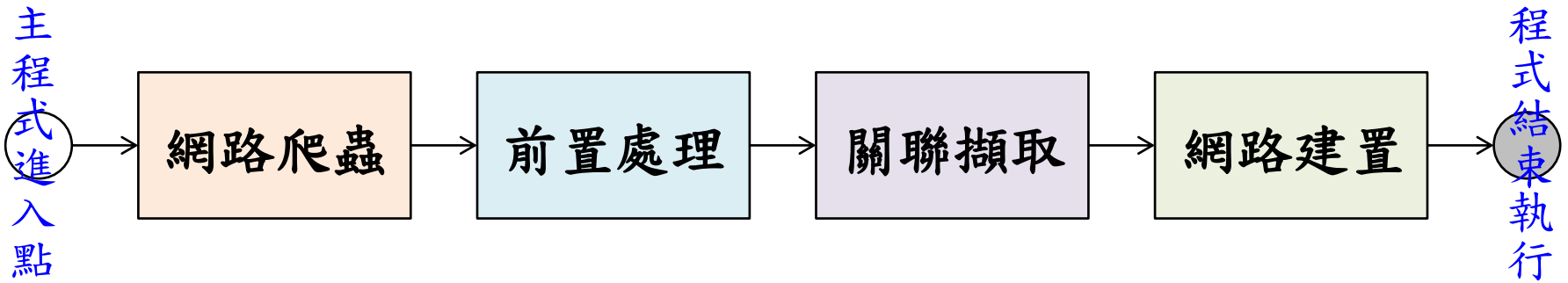
範例1: 預測資料分類




```
if __name__ == '__main__':
    dataPreprocessing('RawTrainingData.txt', 'PrsTrData.txt')
    modelTraining('PrsTrData.txt', 'newModel.bin')
    dataPreprocessing('RawTestData.txt', 'PrsTeData.txt')
    predictResults = classPrediction('PrsTeData.txt', 'newModel.bin')
```

程序導向程式設計 cont.

範例2: 建置基因網路




```
if __name__ == '__main__':
    webCrawler('Liver Cancer', 'rawDataset.xml')
    dataPreprocessing('rawDataset.xml', 'processDataset.txt')
    relationExtraction('processDataset.txt', 'relationDataset.txt')
    networkVisualization('relationDataset.txt')
```

程序導向程式設計 cont.

函式－

可分成自訂函式與內建函式。程式設計師依據需求而自行設計的函式稱為自訂函式，自訂函式必須先定義而後使用。內建的函式可直接使用。

使用關鍵字def進行函式定義。

當函式被呼叫時，須傳入的資料項目。

```
def 函式名稱(參數1, 參數2, ..., 參數n=Value):
    程式敘述I
    程式敘述II
    ...
    return 回傳值
```

函式名稱的命名要有意義
1. 駝峰式名稱命名法
2. 避免使用文字縮寫
3. 須明確指出函式的目的

參數可指定預設值，因此若呼叫函式沒給定引數，參數依然有參考值。

回傳值可以是單一值、多值或物件。

程序導向程式設計 cont.

Python內建函式–

abs()	all()	any()	ascii()	bin()	bool()
bytearray()	bytes()	callable()	chr()	classmethod()	compile()
complex()	delattr()	dict()	dir()	divmod()	enumerate()
eval()	exec()	filter()	float()	format()	frozenset()
getattr()	globals()	hasattr()	hash()	help()	hex()
id()	input()	int()	isinstance()	issubclass()	iter()
len()	list()	locals()	map()	max()	memoryview()
min()	next()	object()	oct()	open()	ord()
pow()	print()	property()	range()	repr()	reversed()
round()	set()	setattr()	slice()	sorted()	staticmethod()
str()	sum()	super()	tuple()	type()	vars()
zip()	__import__()				

程序導向程式設計 cont.

呼叫自訂函式–

若欲指定參數值，則將欲指定參數值的參數放在參數列的最後項。

def functionName(參數$_1$, 參數$_2$, ..., 參數$_n$=Value):

functionName(引數$_1$,引數$_2$, ...,引數$_n$)

呼叫的函式傳入引數，被呼叫的函式接收參數。

functionName(參數$_2$=引數$_2$, 參數$_1$=引數$_1$, ..., 參數$_{n-1}$=引數$_{n-1}$)

透過成對的參數與引數來傳值呼叫某函式。

程序導向程式設計 cont.

範例1: 奇數或偶數

```
if __name__ == '__main__':
    getNumber = 10

    text = isOddOrEven(getNumber)

    print(str(getNumber) + text)
```

```
def isOddOrEven(number):
    description = ''

    if number%2 == 0:
        description = '是偶數'
    else:
        description = '是奇數'

    return description
```

getNumber = 10
輸出: 10是偶數

getNumber = 15
輸出: 15是奇數

程序導向程式設計 cont.

範例2: 費式數列

```
if __name__ == '__main__':
    fibonacciSeries(8)
```

fibonacciSeries(5)
輸出: [1, 1, 2, 3, 5, 8]

fibonacciSeries(8)
輸出: [1, 1, 2, 3, 5, 8, 13, 21, 34]

```
def fibonacciSeries(counts):
    seriesList = []

    for idx in range(0, counts+1):
        if idx <= 1:
            seriesList.append(1)
        else:
            value = seriesList[idx-1] + \
                    seriesList[idx-2]
            seriesList.append(value)

    print(seriesList)
    seriesList.clear()
```

程序導向程式設計 cont.

範例3: 二進制轉十進制

```
if __name__ == '__main__':
    binary2Decimal('10110')
```

```
import math

def binary2Decimal(binaryString):
    number = 0

    for idx in range(len(binaryString)):
        number += pow(2, idx) * \
                  int(binaryString[-1*(idx+1)])

    print(number)
```

binary2Decimal('10110')
輸出: 22

binary2Decimal('01001')
輸出: 9

物件導向程式設計

類別 –

類別為物件設計的模板，物件是由屬性與方法所組成。屬性相當於變數，而方法相當於函式。

EX: 汽車物件

屬性(狀態): 輪胎、方向盤、後照鏡、大燈、保險桿、板金、音響、喇叭、儀表板、...

方法(行為): 煞車防鎖死、車身動態輔助、斜坡起步輔助、車側盲點偵測、巡跡防滑控制、胎壓偵測、...

物件導向程式設計 Cont.

類別具有封裝、繼承與多型的特性。

將所有相關屬性與方法包裝於類別。

```
class 類別名稱(繼承類別):
    def __init__(self, 參數1, 參數2, ..., 參數n):
        self.屬性名稱1 = 參數1
        self.方法名稱(參數2, 參數3)
        ...

    def 方法名稱(self, 參數1, 參數2, ..., 參數n):
        程式敘述I
        程式敘述II
        ...
```

類別建構子，可初始化類別的屬性或方法。

self代表**本類別**，屬性或方法的使用皆要在名稱前加上self。

方法的定義與函式相同，差別在方法的參數項最前方要加上self。

物件導向程式設計 Cont.

範例1: Student Object

```
if __name__ == '__main__':
    myScore = [73, 85, 64]

    stObj = Student('Peter', '1234')
    myAvg = stObj.avgScoring(myScore)

    print('平均分數:' + str(myAvg))
```

```
class Student(object):
    def __init__(self, name, sid):
        self.name = name
        self.sid = sid
        self.scoreDict = {}

    def avgScoring(self, scoreList):
        totalScore = 0

        for score in scoreList:
            totalScore += score

        return totalScore/len(scoreList)
```

屬性定義與初始化

方法定義

myScore = [73, 85, 64]
輸出: 平均分數: 74

myScore = [62, 87, 94, 71]
輸出: 平均分數: 78.5

物件導向程式設計 Cont.

範例2: OnlineShopping Object

```
if __name__ == '__main__':
    myItems = {'Ball':73, 'Pen':85}

    osObj = OnlineShopping('Joe', '4321')
    spend, counts = \
    osObj.totalAmount(myItems)

    print('購買項目:' + str(counts))
    print('購買金額:' + str(spend))
```

```
class OnlineShopping(object):
    def __init__(self, name, uid):
        self.userName = name
        self.userID = uid
        self.itemList = []

    def totalAmount(self, itemDict):
        totalSpend = 0

        for price in itemDict.values():
            totalSpend += price

        return totalSpend,len(itemDict)
```

myItems = {'Ball':73, 'Pen':85}

輸出: 購買項目: 2

購買金額: 158

物件導向程式設計 Cont.

實際案例: 分類問題 – 資料處理

```
class DataProcessing(object):
    # ---------------------------------------------------------------------------------------------- #
    def __init__(self, inputRawFile):
        self.trainData, self.trainLabel, self.testData, self.testLabel = self.datasplit(inputRawFile)
        print('Invoke DataProcessing Class ..')
    # ---------------------------------------------------------------------------------------------- #

    # ---------------------------------------------------------------------------------------------- #
    def datasplit(self, inputFile):
        counter = 0

        iris2class = {'Iris-setosa':0, 'Iris-versicolor':1, 'Iris-virginica':2}
        eachItem, itemList, idxContainer, indiItem = [], [], [], []
        trainData, trainLabel, testData, testLabel = [], [], [], []

        inRaw = open(inputFile, 'r')

        # ================ Read and get the complete raw data =============== #
        for line in inRaw:
            counter += 1
            itemSet = line.split(',')

            for itemValue in itemSet:
                eachItem.append(itemValue.strip())

            itemList.append(eachItem)
            eachItem = []
```

建構子初始化屬性與方法

方法的定義與實作

物件導向程式設計 Cont.

實際案例: 分類問題 – 神經運算

```python
class IrisDetection(object):
    # ------------------------------------------------------------------------------------------------------ #
    def __init__(self, trainData, trainLabel, testData, testLabel):
        self.trainData, self.trainLabel, self.testData, self.testLabel = self.dataPreprocessing(trainData, trainLabel, testData, testLabel)
        print('Invoke IrisIrisDetection Class ..')
    # ------------------------------------------------------------------------------------------------------ #

    # ------------------------------------------------------------------------------------------------------ #
    def dataPreprocessing(self, getTrainData, getTrainLabel, getTestData, getTestLabel):
        getTrainData = np.array(getTrainData)
        getTrainLabel = np.array(getTrainLabel)

        getTestData = np.array(getTestData)
        getTestLabel = np.array(getTestLabel)

        getTrainLabel = np_utils.to_categorical(getTrainLabel)
        getTestLabel = np_utils.to_categorical(getTestLabel)

        return getTrainData, getTrainLabel, getTestData, getTestLabel
    # ------------------------------------------------------------------------------------------------------ #

    # ------------------------------------------------------------------------------------------------------ #
    def buildANN(self):
        model = Sequential()

        model.add(Dense(input_dim=4, units=10, kernel_initializer='normal', activation='relu'))
```

建構子初始化屬性與方法

方法的定義與實作

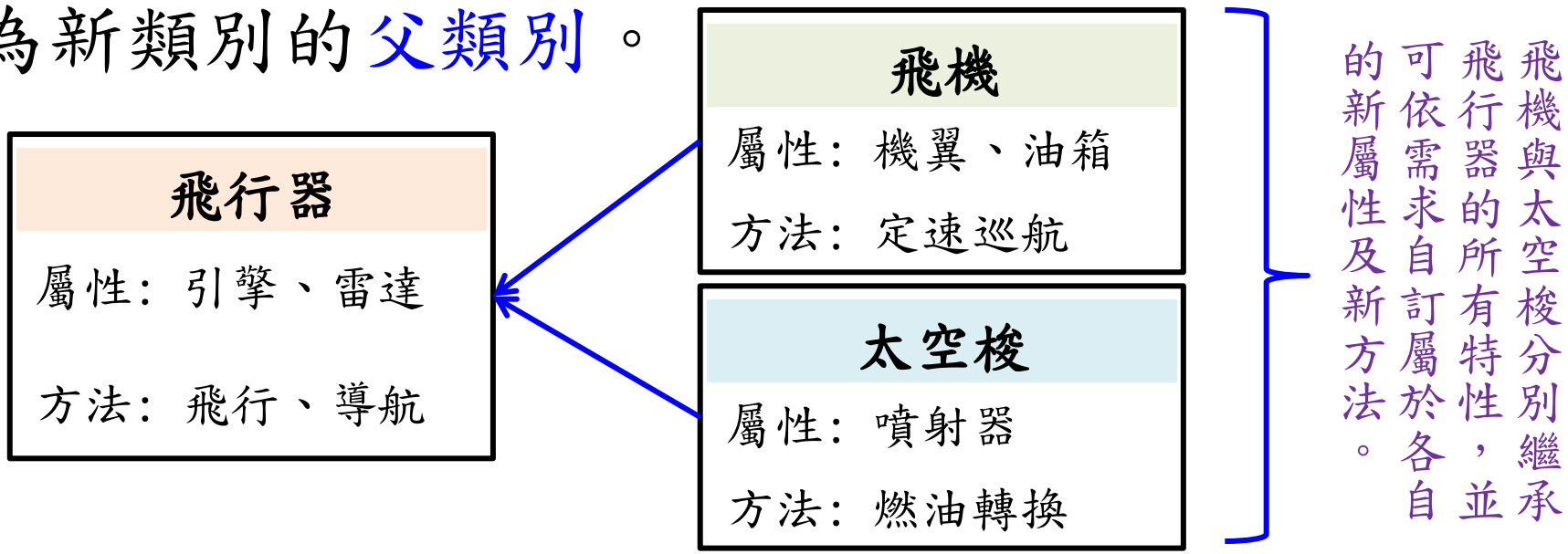
方法的定義與實作

物件導向程式設計 Cont.

繼承 –

新定義的類別可以繼承其他類別。此時新類別具有被繼承類別的所有特性(屬性及方法)。

新類別稱為被繼承類別的子類別，而被繼承類別稱為新類別的父類別。



物件導向程式設計 Cont.

範例1: School Object

```
if __name__ == '__main__':
    schKPI = [81, 79, 92, 86]
    stuScore = [73, 85, 64]

    schObj = School('ABC', 'Oba')
    schObj.name = 'Jonny'
    schObj.sid = '5678'
    schAvg = schObj.avgKPI(schKPI)
    stuAvg = schObj.avgScoring(stuScore)

    print('學校名稱:' + schObj.schoolName)
    print('校長名稱:' + schObj.leader)
    print('學生姓名:' + schObj.name)
    print('學生學號:' + schObj.sid)
    print('績效平均分數:' + str(schAvg))
    print('學生平均分數:' + str(stuAvg))
```

覆寫父類別的屬性值。

繼承Student類別。

```
class School(Student):
    def __init__(self, sName, leader):
        self.schoolName = sName
        self.leader = leader
        self.studentList = []
        self.teacherList = []

    def avgKPI(self, kpiList):
        totalKpi = 0

        for kpiScore in kpiList:
            totalKpi += kpiScore

        return totalKpi/len(kpiList)

    def avgScore(self, scoreList):
        return sum(scoreList)/len(scoreList)
```

覆寫父類別的方法。

物件導向程式設計 Cont.

範例2: Employee Object

```
if __name__ == '__main__':
    psnObj = Person('Mary', 'Lin')
    epyObj = Employee('Joe', 'Liu', '6789')

    print('姓名:' + psnObj.getName())
    print('員工:' + epyObj.employeeInfo())
```

輸出: 姓名: Mary Lin
員工: Joe Liu, 6789

```
class Person(object):
    def __init__(self, first, last):
        self.firstName = first
        self.lastName = last

    def getName(self):
        return self.firstName +' '+ \
               self.lastName
```

```
class Employee(Person):
    def __init__(self, first, last, staffNum):
        Person.__init__(self, first, last)
        self.staffNumber = staffNum

    def employeeInfo(self):
        return self.getName() + ', '+ \
               self.staffNumber
```

亦可用Person.getName(self)取代。

物件導向程式設計 Cont.

多型 –

不同類別可定義相同的方法名稱，各類別亦可針對相同的方法名稱定義各自的功能。程式可藉由呼叫相同的方法名稱，並傳入不同的引數而產生不同的執行結果，此為**多型**的概念。

```
if __name__ == '__main__':
    stringObj = '123456789'
    listObj = ['5', 4, '3', 2, '1', 0]

    print('\'4\'的索引值: ' + str(stringObj.index('4')))
    print('4的索引值: ' + str(listObj.index(4)))
```

輸出如下:

```
'4'的索引值: 3
4的索引值: 1
```

物件導向程式設計 Cont.

內建函式super()可用於呼叫父類別的方法。

範例1: 面積計算

```
class Shape(object):
    def __init__(self, width, height):
        self.width = width
        self.height = height

    def area(self):
        return self.width*self.height
```

```
class Rectangle(Shape):
    def __init__(self, rWidth, rHeight):
        self.width = rWidth
        self.height = rHeight
        self.bgColor = '#6FB1FC'

    def area(self):
        return super().area()
```

```
import math

class Circle(Shape):
    def __init__(self, radius):
        self.radius = radius
        self.pi = 3.14

    def area(self):
        return pow(self.radius, 2)*self.pi
```

```
class Triangle(Shape):
    def __init__(self, tBase, tHeight):
        self.base = tBase
        self.height = tHeight
        self.text = 'Triangle'

    def area(self):
        return 0.5*self.base*self.height
```

程式設計方法

重點整理–

- 程序導向程式設計的架構是建立在所要處理的功能上，依據功能來劃分系統裡的各個模組。讓程式藉由呼叫函式並依循著特定的處理程序來完成所要解決的問題。

- 物件導向程式設計將原來專注於問題的分解，轉換成了解問題的本質，並以物件來將問題模組化。完整的程式功能再由物件所組合而成。物件內含功能的資料屬性和相關方法，可透過訊息的溝通合作以解決實際所面臨的複雜問題。

本週課程終於結束囉^^