Python程式設計

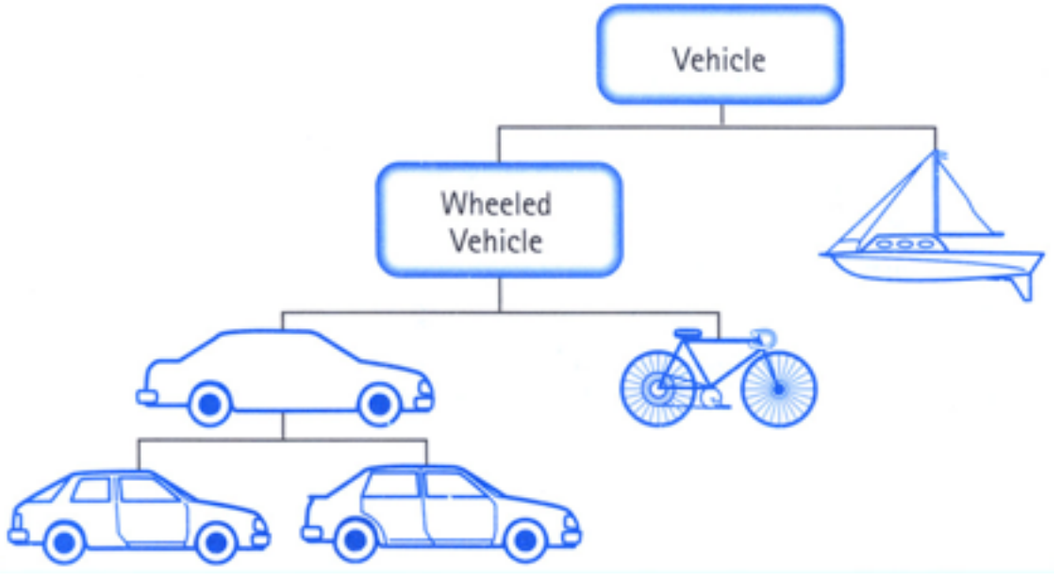
類別繼承(Inherit)

高師大數學系

葉倚任

類別的繼承

- 物件導向中，子類別可以繼承(inherit)父類別，避免重複定義的相同的行為。



類別繼承範例

```
class son:
    surname = 'Yeh'
    hairColor = 'black'
    eye = 'double_eyelid'

    def skill(self):
        print("Play Guitar!")

    def __str__(self):
        return "Surname: {}\nHair color: {}\nEye: {} ".format(
        self.surname,self.hairColor,self.eye)
```

```
class daughter:
    surname = 'Yeh'
    hairColor = 'black'
    eye = 'double_eyelid'

    def skill(self):
        print("Sing!")

    def __str__(self):
        return "Surname: {}\nHair color: {}\nEye: {} ".format(
        self.surname,self.hairColor,self.eye)
```

重複的屬性或者方法
可以利用繼承來避免重複撰寫

類別繼承範例

inherit.py

```
1  class father:
2      surname = 'Yeh'
3      hairColor = 'black'
4      eye = 'double_eyelid'
5
6      def __str__(self):
7          return "Surname: {}\nHair color: {}\nEye: {} ".format(
8          self.surname,self.hairColor,self.eye)
9
10 class son(father):
11     def skill(self):
12         print("Play Guitar!")
13
14 class daughter(father):
15     def skill(self):
16         print("Sing!")
```

重複的屬性或者方法

在子類別中，把父類別當作傳入引數，用以繼承父類別

```
import inherit

myson = inherit.son()
mydaughter = inherit.daughter()
print(myson)
myson.skill()
print('------------------')
print(mydaughter)
mydaughter.skill()
```

```
Surname: Yeh
Hair color: black
Eye:double_eyelid
Play Guitar!
------------------
Surname: Yeh
Hair color: black
Eye:double_eyelid
Sing!
```

練習

```
class SwordsMan:
    def __init__(self, name, level, blood):
        self.name = name   # 角色名稱
        self.level = level # 角色等級
        self.blood = blood # 角色血量

    def fight(self):
        print('揮劍攻擊')

    def __str__(self):
        return "('{name}', {level}, {blood})".format(**vars(self))

    def __repr__(self):
        return self.__str__()
```

```
class Magician:
    def __init__(self, name, level, blood):
        self.name = name   # 角色名稱
        self.level = level # 角色等級
        self.blood = blood # 角色血量

    def fight(self):
        print('魔法攻擊')

    def cure(self):
        print('魔法治療')

    def __str__(self):
        return "('{name}', {level}, {blood})".format(**vars(self))

    def __repr__(self):
        return self.__str__()
```

- 改成用繼承的方式，以避免重複撰寫。

object

- 類別中的實例實際上是由 __new__() 來定義
- 類別中沒有定義 __new__()時，怎麼建構一個實例？

在 Python 中定義一個類別時， 若沒有指定父類別，那麼就是繼承 object 類別

```
print(dir(object))
```

```
['__class__', '__delattr__', '__dir__', '__doc__', '__eq__', '__format__', '__ge__', '__getat
tribute__', '__gt__', '__hash__', '__init__', '__le__', '__lt__', '__ne__', '__new__', '__red
uce__', '__reduce_ex__', '__repr__', '__setattr__', '__sizeof__', '__str__', '__subclasshook_
_']
```

繼承時須注意事項

- 若沒有定義的方法，某些場合下必須呼叫時，就會看看父類別中是否有定義
- 如果定義了自己的方法，那麼就會以你定 義的為主，不會主動呼叫父類別的方法

inherit.py

```
class father:
    surname = 'Yeh'
    hairColor = 'black'
    eye = 'double_eyelid'

    def skill(self):
        print("Compose!")

    def __str__(self):
        return "Surname: {}\nHair color: {}\nEye: {} ".format(
        self.surname,self.hairColor,self.eye)

class son(father):
    def skill(self):
        print("Play Guitar!")

class daughter(father):
    def skill(self):
        print("Sing!")
```

```
import inherit

myson = inherit.son()
mydaughter = inherit.daughter()
myson.skill()
mydaughter.skill()
```

```
Play Guitar!
Sing!
```

父類別的 skill mehtod 被覆蓋了！

利用 super() 呼叫父類別的方法

- 在繼承後，若打算基於父類別的方法實作來重新定義某個方法，可以使用 super() 來呼叫父類別方法

inherit.py

```
1  class father:
2      surname = 'Yeh'
3      hairColor = 'black'
4      eye = 'double_eyelid'
5
6      def skill(self):
7          print("Compose!")
8
9      def __str__(self):
10         return "Surname: {}\nHair color: {}\nEye: {} ".format(
11         self.surname,self.hairColor,self.eye)
12
13 class son(father):
14     def skill(self):
15         super().skill()
16         print("Play Guitar!")
17
18 class daughter(father):
19     def skill(self):
20         super().skill()
21         print("Sing!")
```

利用 super() 呼叫父類別之方法

```
import inherit

myson = inherit.son()
mydaughter = inherit.daughter()
myson.skill()
mydaughter.skill()
```

```
Compose!
Play Guitar!
Compose!
Sing!
```

練習

- 承上一個練習，利用 print 印出物件資料時，也要印出職業類別

多重繼承

- 在 Python 中可以進行多重繼承，也就是一次繼承兩個 父類別的程式碼定義。
- 父類別之間使用逗號作為區隔。

```
class P1:
    def mth1(self):
        print('mth1')

class P2:
    def mth2(self):
        print('mth2')

class S(P1,P2):
    pass

A = S()
A.mth1()
A.mth2()
```

```
mth1
mth2
```

多重繼承之順序

- 如果繼承時多個父類別中有相同的方法名稱，就要注意搜尋的順序。
- 基本上是從子類別開始尋找名稱，接著是同階層父類別由左至右搜尋，再至更上層。
- 同一階層父類別由左至右搜尋，直到達到頂層為止。

```
class C1:
    def mth(self):
        print('C1 mth')

class C2:
    def mth(self):
        print('C2 mth')

class S1(C1,C2):
    pass

class S2(C2,C1):
    pass

A1 = S1()
A2 = S2()
A1.mth()
A2.mth()
```

```
C1 mth
C2 mth
```

利用__mro__ 查詢多重繼承之順序

- 一個子類別在尋找指定的屬性或方法名稱時，會依據類別的 __mro__ 屬性的tuple 中元素順序尋找
- MRO 全名是 Method Resolution Order
- 如果想要知道直接父類別的話，則可以透過類別的__bases__來得知

```
print(S1.__mro__)
```

```
(<class '__main__.S1'>, <class '__main__.C1'>, <class '__main__.C2'>, <class 'object'>)
```

```
print(S1.__bases__)
```

```
(<class '__main__.C1'>, <class '__main__.C2'>)
```

```
print(S2.__mro__)
```

```
(<class '__main__.S2'>, <class '__main__.C2'>, <class '__main__.C1'>, <class 'object'>)
```