**class**

1. Class架構
2. 例子
3. 類別的繼承
4. continue
5. class架構

|  |  |  |
| --- | --- | --- |
| Class class\_name: |  | 定義類別名稱 |
| var\_name  定義  成員變數 |  | 公有的成員變數 |
| \_\_var\_name |  | 私有的成員變數 |
|  |  |  |
| def \_\_init\_\_(self, arg0, arg1, …): |  | 定義建構子 |
| self.成員變數 = arg0  定義成員函式 |  | 設定成員變數 |
| self.成員變數 = arg1 |  | 設定成員變數 |
| …… |  |  |
| def func00(self): |  | 定義公有成員函式func00 |
| self.成員變數 |  | 使用成員變數要有self |
| def func01(self): |  | 定義公有成員函式func01 |
| self.成員變數 |  | 使用成員變數要有self |
| def \_\_func02(self): |  | 定義私有成員函式func02 |
| self.成員變數 |  |  |
|  |  |  |
|  |  |  |

* 1. 類別 -> 抽象的資料型態(data type)
  2. class定義類別名稱
  3. class類別裡面有成員變數與成員函式
  4. **沒底線**表示**公有的**，**雙底線**表示**私有的**，**單底線**表示**保護的**。
  5. **私有的 -> private, 保護的 -> protected, 公有的 -> public。**
  6. 私有的只能在**內部**使用。
  7. 類別名稱宣告類別變數(類別的實例, 也就是物件)
  8. Class裡面的**def**傳入引數都要有**self**
  9. **進入def裡面的成員變數前面皆要加self**
  10. **進入def裡面的成員函數前面皆要加self**
  11. Class宣告**類別變數**(物件)時傳入para進入\_\_init\_\_(self, arg0, arg1, ……)裡面
  12. \_\_init\_\_()可用來**初始化**成員變數
  13. **類別變數.成員變數**，可以用來取用類別變數裡面的成員變數
  14. **類別名稱.成員變數**，可以用來取用各類別變數裡的成員變數，當所有類別變數皆未取用過時，透過**類別名稱.成員變數**可以一次設定所有**類別變數**的**成員變數**。

1. 例子
   1. EX\_00

|  |  |  |
| --- | --- | --- |
| class MyCalClass: |  |  |
| class\_var = 11 |  | 成員變數 |
|  |  |  |
| def \_\_init\_\_(self, x1, y1): |  | 成員函式，建構子 |
| self.x = x1 |  | 初始化成員變數 |
| self.y = y1 |  | 初始化成員變數 |
|  |  |  |
| def calc\_add1(self, a, b): |  | 定義成員函式 |
| self.\_\_calc\_add1() |  | 呼叫私有的成員函式 |
| return a + b |  | 回傳變數 |
|  |  |  |
| def calc\_add2(self): |  | 定義成員函式 |
| return self.x + self.y |  | 回傳成員變數 |
|  |  |  |
| def calc\_mutli(self, a, b): |  | 定義成員函式 |
| print("calc\_add2: ", self.calc\_add2()) |  | 呼叫成員函式，並且列印 |
| return a \* b |  | 回傳變數 |
|  |  |  |
| def calc\_print(self, a): |  | 定義成員函式 |
| print("a value:", a) |  | 列印變數 |
| print("self.y value:", self.y) |  | 列印成員變數 |
|  |  |  |
| def \_\_calc\_add1(self): |  | 定義**私有的**成員函式 |
| print("\_\_calc\_add1") |  |  |
|  |  |  |
| def using\_member\_var(self): |  | 定義成員函式 |
| print("show self.class\_var:", self.class\_var) |  | 列印成員變數 |
| print("show self.calc\_add2();", self.calc\_add2()) |  | 列印成員函式 |
|  |  |  |

* 1. continue

1. 類別的繼承

|  |  |  |
| --- | --- | --- |
| Class class\_name(基礎類別): |  | 定義衍生類別名稱 |
| var\_name |  | 公有的成員變數 |
| \_\_var\_name |  | 私有的成員變數 |
| \_var\_name |  | 保護的成員變數 |
|  |  |  |
| def \_\_init\_\_(self, arg0, arg1, …): |  | 定義建構子 |
| sper().\_\_init\_\_(arg0, arg1, …) |  | 呼叫基礎類別的方法 |
| self.成員變數 = arg0  定義成員函式 |  | 設定成員變數 |
| self.成員變數 = arg1 |  | 設定成員變數 |
| …… |  |  |
| def func00(self): |  | 定義公有成員函式func00 |
| self.成員變數 |  | 使用成員變數要有self |
| super().func00() |  | 呼叫基礎類別func00()的方法 |
| def func01(self): |  | 定義公有成員函式func01 |
| self.成員變數 |  | 使用成員變數要有self |
| def \_\_func02(self): |  | 定義私有成員函式func02 |
| self.成員變數 |  |  |
|  |  |  |
|  |  |  |

* 1. super().父類別的成員函式 -> 呼叫父類別的成員函式。
  2. override(覆蓋) -> 父子類別成員函式有相同的名稱，子override父。
  3. Continue
  4. continue

1. continue