類別的定義與物件的使用

學習目標：

- 了解 Python 類別的定義與物件的使用方式！

類別定義與實作：

1. 類別定義規則：

```
class 類別名稱：
    定義屬性  #其實就是定義變數名稱

    def 方法名稱:
        # 以下就是寫出要做的事
```

PS: __init__(): 類別被呼叫而產生物件時，一開始需要設定的參數內容，亦即物件的初始化方法

self: 物件呼叫自己的自我代稱詞！

__屬性名稱: 表示這是私人的變數，不給外人使用！

__方法名稱: 表示這是私人的方法，不給外人使用！

注意縮排：縮排表示程式區塊！

2. 類別實作範例，Car.py:

```
class Car:
    color = ""
    mark = ""

    def setColor(self,color):
        self.color = color

    def setMark(self,mark):
        self.mark = mark

if __name__ == "__main__":
    myCar = Car()
    myCar.setColor("Red")
    myCar.setMark("Benz")

    print("車子的顏色：", myCar.color)
    print("車子的品牌：", myCar.mark )
```

3. 另一種類別實作範例，Area.py：

```
class Area:
```

```
    # 建構子
    def __init__(self,radius):
        self.radius = radius
        self.PI = 3.141592
        self.area = 0

    # 定義方法
    def getArea(self):
        self.area = self.radius * self.radius * self.PI
        return self.area
```

4. 使用類別範例，demo.py：

```
# 引用類別
import Area
import Car

if __name__ == "__main__":

    # 不要忘記類別的引用方式
    circuleArea = Area.Area(10)
    print("圓面積：",circuleArea.getArea())

    myCar = Car.Car()
    myCar.setColor("White")
    myCar.setMark("BMW")

    print("車子的顏色：", myCar.color)
    print("車子的品牌：", myCar.mark )
```

- 程式可能會有「單獨執行」與「被引用」兩種情形
- 加上 __name__ == '__main__' 的判斷可讓檔案在被引用時，不該執行的程式碼不被執行
- __name__是python中內建、隱含的變數，不必宣告即可用
- 當程式是直接執行時，__name__的值就是__main__
- 當程式是被引用時，__name__的值即是模組名稱

修改遊戲的主程式：

1. 實作玩家類別 Player.py：

```
class Player:
    # 初始化玩家，每人發 20000 遊戲幣以及出發位置為 0
    def __init__(self,money = 20000, po = 0):
        self.__money = money
        self.__po = po

    # 設定玩家名稱
```

```
    def setName(self,name):
        self.__name = name

    # 取得玩家名稱
    def getName(self):
        return self.__name

    # 修改玩家遊戲幣
    def setMoney(self,money):
        self.__money += money

    # 取得玩家遊戲幣
    def getMoney(self):
        return self.__money

    # 修改玩家位置
    def setPo(self,move):
        self.__po += move

    # 取得玩家位置
    def getPo(self):
        return self.__po
```

2. 修改程式主程式 main.py：

```
# 引用 random 類別中的 randrange() 函數
from random import randrange

# 引用 Player 物件
import Player

# 使用 if __name__
if __name__ == "__main__":

    # 要求玩家要輸入遊戲人數
    players_num = eval(input("請輸入玩家人數："))

    # 建立玩家物件
    players = []

    # 按照遊戲人數，使用 Player 類別
    # 逐次產生玩家名稱、玩家代號、玩家初始遊戲幣、玩家初始位置等物件內容
    for i in range(players_num):
        players.append(Player.Player())
        # 要求玩家輸入玩家名稱
        players[i].setName(input("請輸入玩家名稱:"))


    # 輸出資料
    for i in range(players_num):
        print(players[i].getName())
        print(players[i].getPo())
```

```
        print(players[i].getMoney())

    # 設定玩家順序值
    i = 0

    # 開始進行遊戲
    while True:
    ##### a.)
    ##### b.) 擲骰子
        newstep = randrange(1,6)
        print(players[i].getName() + "擲骰子：" + str(newstep) + " 點")
        print(players[i].getName() + "前進中...")
        # 設定玩家新的位置
        players[i].setPo(newstep)
    ##### c.)
    ##### e.)
        # 輪至下一位玩家
        if (i < players_num):
            i = i + 1
        else:
            i = i - players_num

        players_num = players_num - 1 # 測試 while 迴圈用
        print(players_num)

    ##### f.) 結束遊戲條件
        if (players_num <= 1):
            print("遊戲結束")
            break
```

參考文獻：

- 【Python】對付 __pycache__ 的幾種方法
- pyhton中 __pycache__ 文件夾的產生與作用！