PL-Assignment-8

資科三107703004李元亨

1. (30% MI) (a) 請舉例說明the diamond problem in C++，以及C++提供的解決方案。(b)也說明Python (2.2版以後)如何處理此議題。

a.

the diamond problem發生在，當A作為Base， B和C同時繼承A,，然後D又繼承B和C；如果A中存在一個函數B、C有override, 但D沒有的情況，此時compiler就不知道D到底要繼承B, 還是C對函數的定義。
舉例來說:

```
#include <iostream>
#include <string>

using namespace std;

class Animal {
  string m_name;

  public:
  Animal(string n):m_name(n){}
  virtual string name(){return m_name;}
  virtual string talk(){return "wut?";};
};

class Cat : virtual public Animal {
  public:
  Cat(string n):Animal(n){}
  virtual string talk(){return "meow";};
};

class Dog: virtual public Animal {
  public:
  Dog(string n):Animal(n){}
    virtual string talk(){return "woof";};
};

class Monster : virtual public Cat, virtual public Dog {
  public:
  Monster(string n):Cat(n),Dog(n),Animal(n){}
  //virtual string talk(){
  //      return Cat::talk()+Dog::talk()+"waaaa";
  //};
};

int main()
{

  Cat c = Cat("cat");
  cout<<c.name()<<":"<<c.talk()<<endl;

  Dog d("dog");
  cout<<d.name()<<":"<<d.talk()<<endl;

  Monster m("monster");
  cout<<m.name()<<":"<<m.talk()<<endl;
  return 0;
}
```

執行時會遇到diamond problem使得compiler報錯，解決辦法可以再Monster中新增talk的override，其中使用Cat::talk()和Dog::talk()指名上層virtual function，避免talk()面臨ambiguous的問題。

b.

舉例來說:

```
class A:
    def m(self):
        print("m of A called")

class B(A):
    def m(self):
        print("m of B called")

class C(A):
    def m(self):
        print("m of C called")

class D(B,C):
    pass
```

假設d為D的一個instance，呼叫d.m()，Python會按照C3 linearization的方式決定MRO，找到B裡的"print("m of B called")"。

2.(20%, Object) 課堂中提到”Everything is an object in Python”，並以object instantiation為例說明。請另外在舉兩個例子來說明這句話是如何體現在Python的

1.

舉例來說，Python中所有的function都至少有一個attribute **doc**，用來回傳function's source code的doc string。

2.

Python中所有Data type或function都可以被assign成variable,當作argument傳給funciton,當作function的回傳值等等。

3.(30%, JavaScript 6)請閱讀: https://2ality.com/2015/02/es6-classes-final.html接著以JS 6定義課堂中定義過的Point and ColorPoint classes.

```
class Point {
    constructor(x, y) {
        this.x = x;
        this.y = y;
    }
    toString() {
        return '(' + this.x + ', ' + this.y + ')';
    }
    equals( other ){
        console.log("1 equals invoked");
        return( this.x === other.x && this.y === other.y );
    }
}

class ColorPoint extends Point {
    constructor(x, y, color) {
        super(x, y);
        this.color = color;
    }
    toString() {
        return super.toString() + ' in ' + this.color;
    }
    equals( other ){
        console.log("2 equals invoked");
        if( other instanceof ColorPoint ){
            return( super.equals( other ) && this.color === other.color);
        }
        else{
            return super.equals( other );
        }
    }
}

let pt1 = new Point(3,5);
let cp = new ColorPoint(3, 5, 'green');
```

```
let cp2 = new ColorPoint(3, 5, 'red');
let pt2 = cp;

let b = pt2.equals(pt1);
console.log(b);
```

4.(20%)請閱讀以下列文章與其範例，簡要說明closure in Python的運作原理。此部分為期末考主題之一。 https://openhome.cc/eGossip/Blog/UnderstandingLambdaClosure3.html

在文章範例中說明了兩件事，

1. local variable只要還被closure中子函式所reference，在子函式釋放前就不會被釋放
2. 在未使用nonlocal關鍵字的情況下，closure中不能更動上層的local variable

closure這種機制的存在凸顯了python中的function皆是object的事實，function可以透過attribute __closure__ 去紀錄scope內有被reference到的資料。