Lập trình hướng đối tượng

Một số ngôn ngữ hướng đối tượng

Nội dung

Một số đặc điểm hướng đối tượng trong

- Python
- Objective-C

Python

- Được giới thiệu từ 1991
- Ngôn ngữ lập trình bậc cao sử dụng cho nhiều mục đích
- Thuộc dạng thông dịch (interpreted)
- Hỗ trợ nhiều dạng lập trình: lập trình hàm, lập trình thủ tục, lập trình hướng đối tượng

Lớp trong Python

```
class ClassName:
    <statement-1>
    .
    .
    .
    <statement-N>
```

Các phát biểu trong thân lớp thường là những phát biểu định nghĩa phương thức, nhưng cũng có những phát biểu khác

```
class Car:
    pass

c1=Car()
c2=Car()
```

Thêm thuộc tính cho đối tượng

```
class Car:
    pass

c1=Car()
c2=Car()

c1.brand='Toyota'
c2.brand='Ford'
```

Phương thức

- Phương thức được khai báo trong lớp
- Tương tự các hàm

```
class Car:
    def speedUp(self):
        print('Speed up!')

c1=Car()
c1.speedUp()
```

Phương thức khởi tạo

- Phương thức khởi tạo được gọi một cách tự động khi đối tượng được tạo
- Tên __init__ chung cho tất cả các lớp

```
class Car:
    def speedUp(self):
        print('Speed up!')

    def __init__(self):
        print('Initialize a car...')

c1=Car()
c1.speedUp()
```

Phương thức khởi tạo

- Phương thức khởi tạo có thể có thêm tham số
- Các thuộc tính của đối tượng thuộc lớp thường được khai báo trong phương thức này

```
class Car:

    def __init__(self, s):
        self.brand=s
        print('Initialize a '+self.brand+' car...')

c1=Car('Toyota')
print(c1.brand)
```

Thêm thuộc tính cho đối tượng

```
class Car:

    def __init__(self, s):
        self.brand=s
        print('Initialize a '+self.brand+' car...')

    def stop(self):
        self.speed=0
        print('Stopped.')



c1=Car('Toyota')
print(c1.brand)
#c1.stop()
print(c1.speed)
```

Thuộc tính lớp

- Tương tự thuộc tính *static* trong Java

```
class Car:
    number_of_cars=0
    def __init__(self, s):
        Car.number_of_cars=Car.number_of_cars+1
        self.brand=s
        print('Initialize a '+self.brand+' car...')


c1=Car('Toyota')
c2=Car('Ford')
print(Car.number_of_cars)
```

public, protected, và private?

- Tất cả các thuộc tính và phương thức đều có thể truy xuất được từ bên ngoài
- "Quy ước":
  - _m: để thể hiện m là thuộc tính, phương thức dạng *protected*
  - __m: để thể hiện m là thuộc tính, phương thức dạng *private*

public, protected, và private?

```
class Car:

    def __init__(self, s):
        self.brand=s
        print('Initialize a '+self.brand+' car...')

    def _maxSpeed(self):
        print('As fast as I can!')

    def __destroy(self):
        print('Bye!')

c1=Car('Toyota')
c1._maxSpeed() #ok
c1.__destroy() #error
c1._Car__destroy() #ok
```

Kế thừa

- Python hỗ trợ kế thừa

```
class SubClass(SuperClass)
```

```
class Car:
    number_of_cars=0
    def __init__(self, s):
        Car.number_of_cars=Car.number_of_cars+1
        self.brand=s
        print('Initialize a '+self.brand+' car...')


c1=Car('Toyota')
c2=Car('Ford')
print(Car.number_of_cars)
```

```
class ECar(Car):
    pass
```

Tạo đối tượng của lớp con

```
class Car:
    number_of_cars=0
    def __init__(self, s):
        Car.number_of_cars=Car.number_of_cars+1
        self.brand=s
        print('Initialize a '+self.brand+' car...')


c1=Car('Toyota')
c2=Car('Ford')
print(Car.number_of_cars)


class ECar(Car):
    pass

e1=ECar('Tesla')
print(Car.number_of_cars)  #3
print(ECar.number_of_cars) #3
```

Override và super

```
10 class Car:
11     number_of_cars=0
12     def __init__(self, s):
13         Car.number_of_cars=Car.number_of_cars+1
14         self.brand=s
15         print('Initialize a '+self.brand+' car...')
16 
17 
18 c1=Car('Toyota')
19 c2=Car('Ford')
20 print(Car.number_of_cars)
21 
22 
23 class ECar(Car):
24     number_of_ecars=0
25     def __init__(self,s):
26         super().__init__(s)
27         ECar.number_of_ecars=ECar.number_of_ecars+1
28 
29 e1=ECar('Tesla')
30 print(Car.number_of_cars)   #3
31 print(ECar.number_of_ecars) #1
```

Overloading

```
def speed_up(self):
    print('Speed up')

def speed_up(self,speed):
    print('Speed up to '+speed)
```

- Phương thức khai báo trước sẽ bị "che" (không sử dụng được)
- Giải pháp chính là sử dụng giá trị mặc định

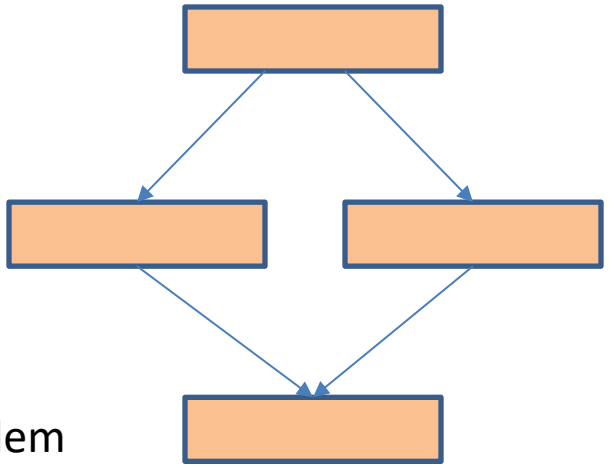
```
def speed_up(self,speed=None):
    if speed is None:
        print('Speed up')
    else:
        print('Speed up to '+speed)
```

Đa kế thừa

- Python cho phép đa kế thừa

```
class SubClass(SuperClass1, SuperClass2)
```

- Nên hạn chế tối đa việc sử dụng đa kế thừa



Diamond problem

Đa hình

- Python hỗ trợ đa hình
- Tuy nhiên, với đặc điểm của ngôn ngữ là “duck typing”, tính đa hình không còn quá quan trọng

Đa hình

```
43 class Vehicle:
44     def move(self):
45         pass
46 
47 class Car(Vehicle):
48     def move(self):
49         print('Running...')
50 class Airplane(Vehicle):
51     def move(self):
52         print('Flying...')
53 
54 
55 mlist=[]
56 
57 mlist.append(Vehicle())
58 mlist.append(Car())
59 mlist.append(Airplane())
60 
61 for e in mlist:
62     e.move()
```

Đa hình

```
43 class Vehicle:
44     def move(self):
45         pass
46 
47 class Car(Vehicle):
48     def move(self):
49         print('Running...')
50 class Airplane(Vehicle):
51     def move(self):
52         print('Flying...')
53 
54 class Baby:
55     def move(self):
56         print('Crawling...')
57 
58 mlist=[]
59 
60 mlist.append(Vehicle())
61 mlist.append(Car())
62 mlist.append(Airplane())
63 mlist.append(Baby())
64 
65 for e in mlist:
66     e.move()
```

Objective-C

- Được đưa vào những năm đầu 1980
- Dựa trên ngôn ngữ C
- Là ngôn ngữ bậc cao sử dụng cho nhiều mục đích
- Được xây dựng theo dạng hướng đối tượng

Lớp trong Objective-C

```
2  @interface NewClassName: ParentClass {
3
4
5  ClassMembers;
6  }
7
8  ClassMethods;
9  @end
```

```
3  @interface BankAccount: NSObject
4  {
5
6  }
7
8  @end
9
```

Thuộc tính

```
2
3  @interface BankAccount: NSObject
4  {
5          double accountBalance;
6          long accountNumber;
7  }
8
9  @end
10
```

Khai báo phương thức

```
3  @interface BankAccount: NSObject
4  {
5          double accountBalance;
6          long accountNumber;
7  }
8  -(void) setAccount: (long) y andBalance: (double) x;
9  -(void) setAccountBalance: (double) x;
10 -(double) getAccountBalance;
11 -(void) setAccountNumber: (long) y;
12 -(long) getAccountNumber;
13 -(void) displayAccountInfo;
14 @end
15
```

Định nghĩa phương thức

```
3  @implementation NewClassName
4
5  ClassMethods
6
7  @end
8
```

```
3  @implementation BankAccount
4
5  -(void) setAccount: (long) y andBalance: (double) x;
6  {
7          accountBalance = x;
8          accountNumber = y;
9  }
10 -(void) setAccountBalance: (double) x
11 {
12         accountBalance = x;
13 }
14
15 -(double) getAccountBalance
16 {
17         return accountBalance;
18 }
19
20 @end
```

Overloading

- Objective-C không hỗ trợ overloading
- Giải pháp chính

```
9   -(void) setAccount: (long) y;
10  -(void) setAccount: (long) y andBalance: (double) x;
```

Overriding

- Objective-C có hỗ trợ overriding
- Dynamic binding được hỗ trợ thông qua kiểu "id"

```
3   B *b;
4   C *c;
5
6   id obj;
7
8   obj=b;
9   [obj m];
10
11  obj=c;
12  [obj m];
```

Đa kế thừa

- Không hỗ trợ