TRƯỜNG ĐẠI HỌC VINH
VIỆN KỸ THUẬT VÀ CÔNG NGHỆ

CHƯƠNG 6

LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

Nghệ An, 2021

Chương 6:
LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

NỘI DUNG GIẢNG DẠY:

6.1. Lập trình hướng đối tượng

6.2. Các khái niệm cơ bản

6.3. Lớp và đối tượng

6.4. Thuộc tính

6.5. Phương thức

6.6. Kế thừa

6.7. Đa hình

6.8. Các phương thức đặc biệt

LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG TRONG PYTHON

Lập trình hướng đối tượng (gọi tắt là OOP, từ chữ Anh ngữ object-oriented programming), là kĩ thuật lập trình hỗ trợ công nghệ đối tượng. OOP được xem là giúp tăng năng suất, đơn giản hóa độ phức tạp khi bảo trì cũng như mở rộng phần mềm bằng cách cho phép lập trình viên tập trung vào các đối tượng phần mềm ở bậc cao hơn. Ngoài ra, nhiều người còn cho rằng OOP dễ tiếp thu hơn cho những người mới học về lập trình hơn là các phương pháp trước đó (theo Wikipedia).

LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG TRONG PYTHON

- **Các khái niệm cơ bản:**

  Trừu tượng (Abstraction)

  Đa hình (Polymorphism)

  Đóng gói (Encapsulation)

  Kế thừa (Inheritance)

LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG TRONG PYTHON

- **Khái niệm đối tượng (Object) :**

Đối tượng là các thực thể của một lớp nào đó. Lớp là khuôn mẫu của các đối tượng. Ví dụ như list, tuple, dictionary, string, int… là các lớp. Khi chúng ta khai báo biến thuộc các lớp này thì chúng là các đối tượng. Tất cả mọi thứ trong Python đều là đối tượng.

LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG TRONG PYTHON

- **Khái niệm đối tượng (Object) :**

```
import sys

def function(): pass

print (type(1))
print (type(""))
print (type([]))
print (type({}))
print (type(()))
print (type(object))
print (type(function))
print (type(sys))
```

Trong ví dụ trên nhờ vào hàm type() mà chúng ta biết được thực chất tất cả các kiểu dữ liệu và các module mà chúng ta đã học thực chất đều là các đối tượng.

LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG TRONG PYTHON

- **Lớp (Class) và Đối tượng (Object)**:

- Đối tượng (Object) là những thực thể tồn tại có hành vi. Ví dụ đối tượng là một xe ô tô có tên hãng, màu sắc, loại nguyên liệu, hành vi đi, dừng, đỗ, nổ máy...

- Lớp (Class) là một kiểu dữ liệu đặc biệt do người dùng định nghĩa, tập hợp nhiều thuộc tính đặc trưng cho mọi đối tượng được tạo ra từ lớp đó.

Thuộc tính là các giá trị của lớp. Sau này khi các đối tượng được tạo ra từ lớp, thì thuộc tính của lớp lúc này sẽ trở thành các đặc điểm của đối tượng đó.

LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG TRONG PYTHON

- **Phân biệt giữa Đối tượng (Object) và Lớp (Class):**

- Đối tượng (Object): có trạng thái và hành vi.

- Lớp (Class): có thể được định nghĩa như là một template mô tả trạng thái và hành vi mà loại đối tượng của lớp hỗ trợ.

Một đối tượng là một thực thể (instance) của một lớp.

Một lớp có thể hiểu là một bản thiết kế để tạo ra một thực thể (mà chúng ta gọi là đối tượng). Một đối tượng là một thực thể được xây dựng từ một bản thiết kế đó (mà chúng ta gọi là lớp). Ví dụ chúng ta định nghĩa lớp Dog, thì chúng ta có các đối tượng là Huck, Lulu….

LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG TRONG PYTHON

- **Phân biệt giữa Đối tượng (Object) và Lớp (Class):**

LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG TRONG PYTHON

- **Phân biệt giữa Đối tượng (Object) và Lớp (Class):**

```
class First:
    pass

fr = First()

print (type(fr))
print (type(First))
```

LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG TRONG PYTHON

- **Thuộc tính:**

Thuộc tính là biến nằm trong một lớp. Thuộc tính mô tả các đặc tính của một đối tượng. Trong Python có một phương thức đặc biệt gọi là __init__() dùng để khởi tạo giá trị cho các thuộc tính của một đối tượng.

LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG TRONG PYTHON

- **VD Thuộc tính:**

```
class Cat:
    def __init__(self, name):
        self.name = name

missy = Cat('Missy')
lucky = Cat('Lucky')

print (missy.name)
print (lucky.name)
```

LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG TRONG PYTHON

- **Chú ý về thuộc tính:**

Có thể gán giá trị cho các thuộc tính ở bất cứ đâu sau phần định nghĩa lớp chứ không chỉ riêng bên trong phương thức khởi tạo.

```
class Dynamic:
    pass

d = Dynamic()
d.name = "Dynamic"
print (d.name)
```

LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG TRONG PYTHON

- **Chú ý về thuộc tính:**

Một đặc điểm nữa trong Python bạn có thể định nghĩa các thuộc tính chung cho mọi đối tượng (**ATTRIBUTE)**.

Có hai cách để truy xuất thuộc tính lớp, thứ nhất là thông qua tên lớp, cách thứ hai là thông qua một thuộc tính đặc biệt nữa là thuộc tính __class__.

```
class Cat:
  species = ‘dongvat'

  def __init__(self, name, age):
    self.name = name
    self.age = age


missy = Cat('Missy', 3)
lucky = Cat('Lucky', 5)

print (missy.name, missy.age)
print (lucky.name, lucky.age)

print (Cat.species)
print (missy.__class__.species)
print (lucky.species)
```

LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG TRONG PYTHON

- **Phương thức:**

Phương thức là các hàm, sự khác biệt ở chỗ chúng được định nghĩa bên trong một lớp. Các phương thức được sử dụng để thực hiện các công việc cụ thể.

LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG TRONG PYTHON

- **Phương thức:**

```
class Circle:
    pi = 3.141592

    def __init__(self, radius=1):
        self.radius = radius

    def area(self):
        return self.radius * self.radius * Circle.pi

    def setRadius(self, radius):
        self.radius = radius

    def getRadius(self):
        return self.radius
```

```
c = Circle()

c.setRadius(5)
print (c.getRadius())
print (c.area())
```

LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG TRONG PYTHON

- **Kế thừa:**

Kế thừa là định nghĩa một lớp dựa trên một lớp đã được định nghĩa trước đó. Lớp kế thừa từ lớp khác được gọi là lớp dẫn xuất, lớp được các lớp khác kế thừa mình thì gọi là lớp cơ sở. Kế thừa trong lập trình hướng đối tượng cho phép chúng ta sử dụng lại mã nguồn và giảm độ phức tạp của chương trình. Lớp dẫn xuất có thể kế thừa hoặc mở rộng các tính năng của lớp cơ sở.

LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG TRONG PYTHON

- **Kế thừa:**

```
class Animal:
    def __init__(self):
        print ("Animal created")

    def whoAmI(self):
        print ("Animal")

    def eat(self):
        print ("Eating")
```

```
class Dog(Animal):
    def __init__(self):
        Animal.__init__(self)
        print ("Dog created")

    def whoAmI(self):
        print ("Dog")

    def bark(self):
        print ("Woof!")


d = Dog()
d.whoAmI()
d.eat()
d.bark()
```

LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG TRONG PYTHON

- **Kế thừa:**

Để kế thừa một lớp thì chúng ta đặt tên lớp đó bên trong cặp dấu ngoặc tròn () ngay phía sau phần định nghĩa tên lớp. Nếu bên trong lớp cơ sở đã định nghĩa phương thức __init__(), chúng ta phải gọi lại phương thức __init__() từ lớp cơ sở.

```
class Dog(Animal):
    def __init__(self):
        Animal.__init__(self)
        print ("Dog created")
```

LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG TRONG PYTHON

- **Đa hình:**

Đa hình là việc có thể sử dụng các toán tử hay các hàm trên nhiều kiểu dữ liệu khác nhau.

VD:

```
a = "alfa"
b = (1, 2, 3, 4)
c = ['o', 'm', 'e', 'g', 'a']

print (a[2])
print (b[1])
print (c[3])
```

LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG TRONG PYTHON

- **Đa hình:**

```
class Animal:
    def __init__(self, name=''):
        self.name = name

    def talk(self):
        pass


class Cat(Animal):
    def talk(self):
        print ("Meow!")
```

```
class Dog(Animal):
    def talk(self):
        print ("Woof!")


a = Animal()
a.talk()

c = Cat("Missy")
c.talk()

d = Dog("Rocky")
d.talk()
```

LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG TRONG PYTHON

- **Đa hình:**

Trong ví dụ trên, định nghĩa hai lớp chó (Dog) và mèo (Cat) kế thừa từ lớp Animal. Do đó cả hai lớp này đều kế thừa phương thức talk() của lớp Animal, nhưng mỗi lớp lại in ra hai dòng text khác nhau.

LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG TRONG PYTHON

- **Các phương thức đặc biệt:**

Tất cả các lớp dù là có sẵn hay do chúng ta định nghĩa đều kế thừa từ một lớp gốc trong Python có tên là object. Lớp này có sẵn một số phương thức và đương nhiên là các lớp do chúng ta định nghĩa đều kế thừa các phương thức này, ví dụ như phương thức __init__()… Trong Python khi chúng ta gọi đến các hàm hay toán tử được xây dựng sẵn như print(), del… chúng sẽ gọi đến các phương thức gốc của lớp object. Chính vì các lớp do chúng ta định nghĩa đều được kế thừa từ lớp object nên chúng ta cũng có thể dùng các hàm hay toán tử có sẵn trong Python với các lớp của chúng ta.

LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG TRONG PYTHON

- **Các phương thức đặc biệt:**

```
class Vector:

  def __init__(self, data):
    self.data = data

  def __str__(self):
    return repr(self.data)

  def __add__(self, other):
    data = []
    for j in range(len(self.data)):
      data.append(self.data[j] + other.data[j])
    return Vector(data)
```

```
  def __sub__(self, other):
    data = []
    for j in range(len(self.data)):
      data.append(self.data[j] - other.data[j])
    return Vector(data)
```

```
x = Vector([1, 2, 3])
y = Vector([3, 0, 2])
print (x + y)
print (y - x)
```

BÀI TẬP

NỘI DUNG:

1. Định nghĩa class Nguoi và 2 class con của nó: Nam, Nu. Tất cả các class có method "getGender" có thể in "Nam" cho class Nam và "Nữ" cho class Nu. .

2. Định nghĩa một class có tên là Shape và class con là Square. Square có hàm init để lấy đối số là chiều dài. Cả 2 class đều có hàm area để in diện tích của hình, diện tích mặc định của Shape là 0.

BÀI TẬP

CHUẨN BỊ CHO BUỔI HỌC TIẾP THEO:

1. Đọc các tài liệu về nội dung mục 7.1; 7.2.

2. Tìm hiểu khái niệm về tập tin và thư mục trong hệ điều hành.