Python 3x Programing

Lesson Python 3.x

Tutor MR. Huynh Nam

Time 90 mins

Ai-Edu

1

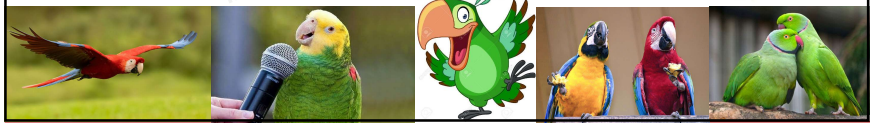
Content

- Introduction to OOP in Python
- Python Class
- Python Inheritance
- Multiple Inheritance
- Operator Overloading
- Example

2

Introduction to OOP in Python

- Python is a multi-paradigm programming language. Meaning, it supports different programming approach.
- One of the popular approach to solve a programming problem is by creating objects. This is known as Object-Oriented Programming (OOP).
- An object has two characteristics:
  - Attributes
  - Behavior
- Example: Parrot is an object,
  - Name, age, color are **attributes**
  - Flying, singing, dancing, eating and love are **behavior**



3

The Concept of OOP

- Abstraction
- Inheritance: A process of using details from a new class without modifying existing class.
- Encapsulation: Hiding the private details of a class from other objects.
- Polymorphism: A concept of using common operation in different ways for different data input.

4

Example in mind

5

Class

- A class is a blueprint for the object.
- Use class keyword to define a class.
- From class, we construct instances.
- An instance is a specific object created from a particular class.

7

Example Class Parrot

```
class Parrot:
    # instance attribute
    def __init__(self, name, age):
        self.name = name
        self.age = age
```

```
class Parrot:
    pass
```


8

Object

- An object (instance) is an instantiation of a class.
- When class is defined, only the description for the object is defined. Therefore, no memory or storage is allocated.
- The example for object of parrot class can be:

```
class Parrot:
    pass
```

1

```
obj = Parrot()
```

```
class Parrot:
    # instance attribute
    def __init__(self, name, age):
        self.name = name
        self.age = age
```

2

```
# instantiate the Parrot class
blu = Parrot("Blu", 10)
woo = Parrot("Woo", 15)

# access the instance attributes
print("{} is {} years old".format( blu.name, blu.age))
print("{} is {} years old".format( woo.name, woo.age))
```


9

Class Attribute

```
class Parrot:

    # class attribute
    species = "bird"

    # instance attribute
    def __init__(self, name, age):
        self.name = name
        self.age = age
```

```
# instantiate the Parrot class
blu = Parrot("Blu", 10)
woo = Parrot("Woo", 15)

# access the class attributes
print("Blu is a {}".format(blu.__class__.species))
print("Woo is also a {}".format(woo.__class__.species))

# access the instance attributes
print("{} is {} years old".format( blu.name, blu.age))
print("{} is {} years old".format( woo.name, woo.age))
```

```
Blu is a bird
Woo is also a bird
Blu is 10 years old
Woo is 15 years old
```


10

Methods

- Methods are functions defined inside the body of a class. They are used to define the behaviors of an object.

```
class Parrot:

    # instance attributes
    def __init__(self, name, age):
        self.name = name
        self.age = age

    # instance method
    def sing(self, song):
        return "{} sings {}".format(self.name, song)

    def dance(self):
        return "{} is now dancing".format(self.name)
```

```
# instantiate the object
blu = Parrot("Blu", 10)

# call our instance methods
print(blu.sing("'Happy'"))
print(blu.dance())
```

```
Blu sings 'Happy'
Blu is now dancing
```

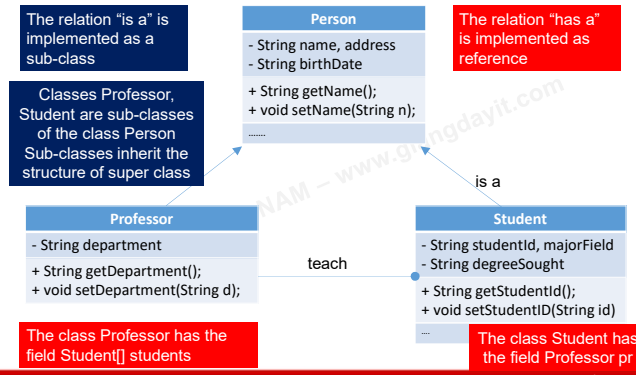

11

Example - 1

- Point in plane (**Point2D**) is defined as: coordinator X (**x**), coordinator Y (**y**), show 2D-point information (**display**), calculate distance between two 2D-point (**distance**) based on Euclid's formulas
- Design OOP Point2D class which has field, method, constructor.
- Creating two instances as **pointA**, **pointB**. Then showing their information
- Display on screen the distance of two points above

12

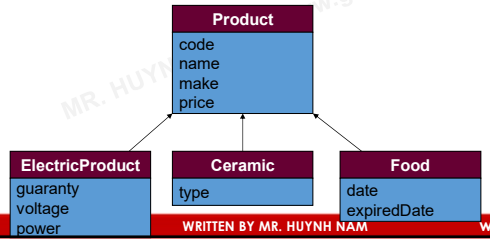
Concept of Inheritance



15

Inheritance

- Inheritance is a way of creating new class for using details of existing class without modifying it. The newly formed class is a derived class (or child class). Similarly, the existing class is a base class (or parent class).
- Example:
  - Electric Products< code, name, make, price, guaranty, voltage, power>
  - Ceramic Products < code, name, make, price, type >
  - Food Products < code, name, make, price , date, expiredDate >

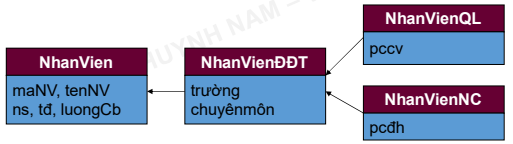


16

Example in mind - Vietnamese

Phân tích phân cấp thừa kế cho các đối tượng trong một học viện:

- ***Nhân viên quản lý*** <mã nv, tên nv, năm sinh, trình độ, trường đào tạo, chuyên môn, lương cb, phụ cấp chức vụ>
- ***Nhân viên nghiên cứu*** <mã nv, tên nv, năm sinh, trình độ, trường đào tạo, chuyên môn, lương cb, phụ cấp độc hại>
- ***Nhân viên phục vụ*** <mã nv, tên nv, năm sinh, trình độ, lương cb>



17

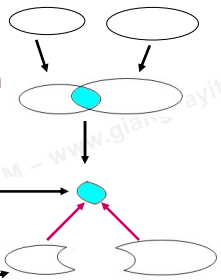
Technique in Inheritance - Vietnamese

(1) Liệt kê đặc điểm của các đối tượng cần quan tâm.

(2) Tìm tập giao của các tính chất giữa các lớp, tách tập giao này để xây dựng lớp cha.

(3) Đặt 1 tên gọi có ý nghĩa cho lớp cha.

(4) Phần còn lại sau khi tách tập giao là các lớp con.



18

Example

- Encapsulation
- Inheritance
- Polymorphism

19

Exercise

- Đối tượng hình chữ nhật (**Rectangle**) bao gồm các thuộc tính về chiều dài (**length**), chiều rộng (**width**) và phương thức tính diện tích (**area**)
- Đối tượng hình hộp (**Box**) cũng bao gồm các thuộc tính về chiều dài (**length**) và chiều rộng (**width**). Tuy nhiên, có bổ sung them thuộc tính chiều cao (**height**). Ngoài ra có 2 phương thức tính diện tích xung quanh(**area**) và tính thể tích (**volumn**)
- Hãy vẽ sơ đồ thiết kế hướng đối tượng cho mô tả trên bao gồm: **attributes, constructors, methods, getters, setters**
- Hãy viết chương trình nhập danh sách các hình và xuất ra trong danh sách đó có bao nhiêu hình chữ nhật và bao nhiêu hình hộp

23

Conclusion

- The programming gets easy and efficient.
- The class is sharable, so codes can be reused.
- The productivity of programmars increases
- Data is safe and secure with data abstraction.

24

THANK YOU
Q & A

25