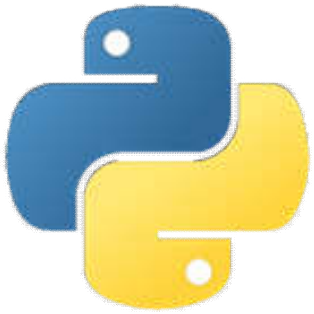


Kế thừa & Đa hình trong Python

Nội dung

1. Nhắc lại bài cũ
2. Kế thừa
3. Đa kế thừa
4. Đa hình
5. Bài tập thực hành

Nhắc lại bài cũ

- Class – nguyên mẫu tạo ra đối tượng
- Attribute – thuộc tính của đối tượng
- Method – phương thức của đối tượng
- Constructor – phương thức khởi tạo đối tượng

Kế thừa

- **Inheritance** - **Python** cho phép tạo một lớp mở rộng từ một hoặc nhiều lớp khác. Lớp này được gọi là lớp dẫn xuất (Derived class) hoặc đơn giản gọi là lớp con.
- Lớp con sẽ thừa kế các thuộc tính (attribute), phương thức và các thành viên khác từ lớp cha.
- Nếu lớp con không tự định nghĩa 1 constructor của nó, mặc định nó sẽ thừa kế constructor của lớp cha.

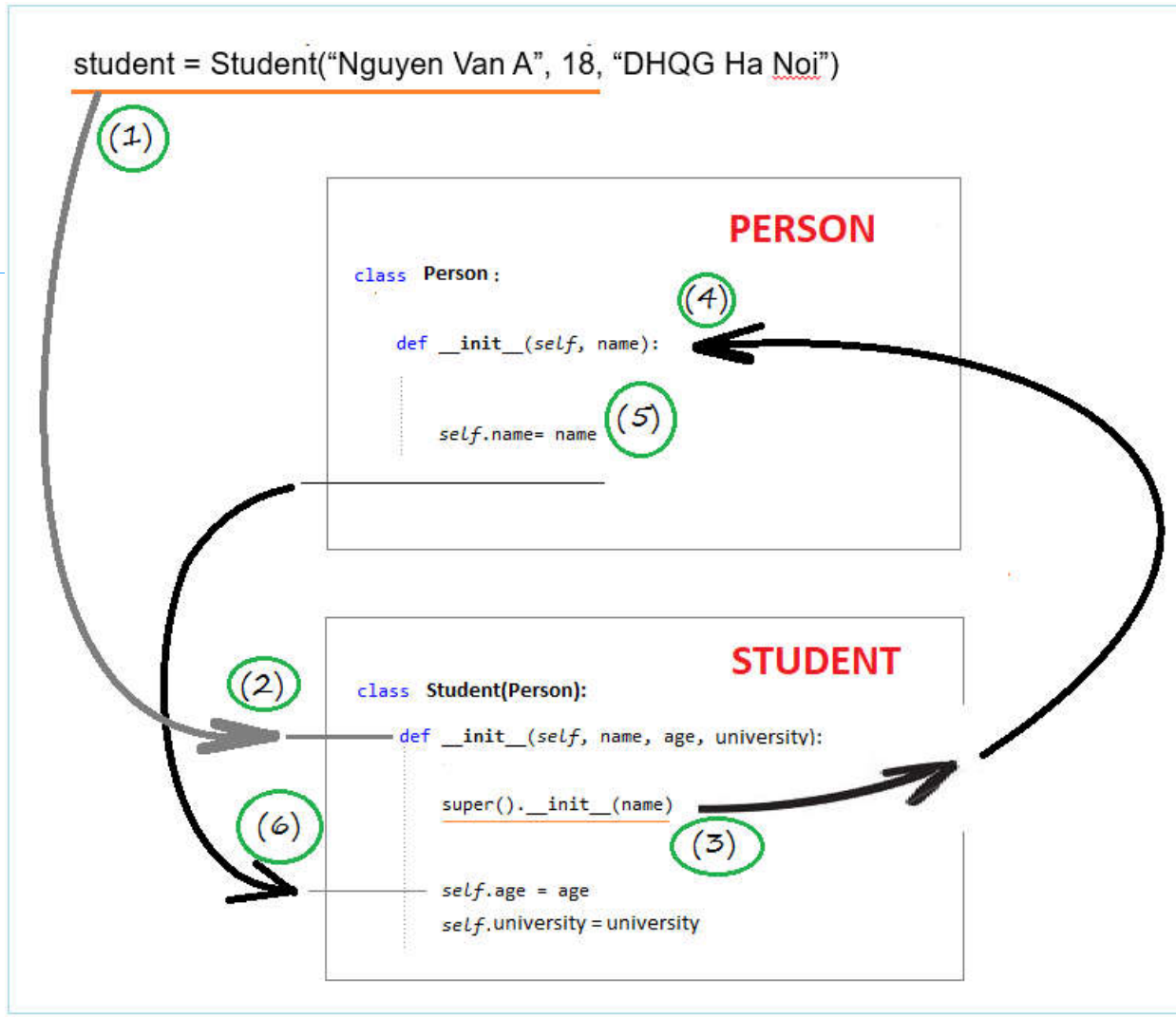
❑ Cú pháp kế thừa:

```
from classparent import ClassParent

# ClassChildren là lớp dẫn xuất hay lớp con
# ClassParent là lớp cha hay lớp được thừa kế

class ClassChildren(ClassParent):

  # Code ...
```

```
class Person:
    # Hàm tạo của lớp Person
    def __init__(self, name):

        # Lớp Person có 1 thuộc tính (attribute): 'name'.
        self.name= name

    # Phương thức (method):
    def showInfo(self):
        print ("This ‘s" + self.name)

    # Phương thức (method):
    def move(self):
        print ("moving ...")
```

- Phương thức khởi tạo (constructor) của lớp con bao giờ cũng gọi tới phương thức khởi tạo của lớp cha để gán giá trị cho các thuộc tính của lớp cha, sau đó nó mới gán giá trị cho các thuộc tính của nó.

```
from person import Person

# Lớp Student (Sinh viên) mở rộng (extends) từ lớp Person.
class Student (Person):
    def __init__(self, name, age, university):
        # Gọi tới constructor của lớp cha (Person) để gán giá trị vào
thuộc tính 'name' của lớp cha.
        super().__init__(name)
        self.age = age
        self.university = university
```


student = Student("Nguyen Van A", 18, "DHQG Ha Noi")
(1)
PERSON
class Person :
(4)
def __init__(self, name):
self.name= name
(5)
STUDENT
(2)
class Student(Person):
def __init__(self, name, age, university):
super().__init__(name)
(6)
(3)
self.age = age
self.university = university

- Khởi tạo đối tượng Studend và gọi đến hàm showInfo. Mặc dù lớp Student không có hàm showInfo, nhưng kế thừa từ lớp cha Person.

```
from student import Student

# Khởi tạo đối tượng Student
student = Student(“Nguyen Van A”, 18, “DHQG Ha Noi”)

# Gọi tới hàm showInfo của lớp cha (Person) qua đối tượng Student
student.showInfo()
```

Bài tập kế thừa

Bài 1:

- a) Tạo lớp Animal có các thuộc tính name (tên), age (tuổi) và weight (cân nặng) và phương thức là showInfo(), phương thức này dùng để hiển 3 thông tin tên, tuổi, cân nặng.
- b) Tạo lớp Mouse kế thừa lớp Animal.
- c) Tạo một file main.py, thực hiện tạo đối tượng Mouse1 trên file main với các tham số truyền vào là tên = "Jerry", tuổi = 2, cân nặng = 0,1Kg. Sau đó gọi đến hàm showInfo() để hiển thị 3 thông tin trên.

Override

- Override – cho phép lớp con có thể ghi đè các phương thức từ class cha.

```
from person import Person

# Lớp Student (Sinh viên) mở rộng (extends) từ lớp Person.
class Student (Person):
    def __init__(self, name, age, university):
        # Gọi tới constructor của lớp cha (Person) để gán giá trị vào thuộc tính 'name' của lớp cha.
        super().__init__(name)
        self.age = age
        self.university = university

    # Ghi đè (override) phương thức cùng tên của lớp cha.
    def showInfo(self):
        print ("Name: " + self.name)
        print (" Age: " + str(self.age))
        print (" University:  " + self.university)
```

- Khởi tạo đối tượng Studend và gọi đến hàm showInfo. Lúc này giá trị trả về là hàm showInfo của lớp con Student chứ không phải của lớp cha Person.

```
from student import Student

# Khởi tạo đối tượng Student
student = Student(“Nguyen Van A”, 18, “DHQG Ha Noi”)

# Gọi tới hàm showInfo của lớp Student
student.showInfo()
```

Bài tập override

Bài 1 (tiếp):

- d) Bổ sung vào lớp Mouse đã tạo 2 thuộc tính là country (quê hương), year (năm sản xuất).
- e) Tạo hàm showInfo() tại lớp Mouse. Hàm này để hiển thị thông tin bao gồm tên, quê hương và năm sản xuất.
- f) Tại file main.py đã có ở mục trước, thực hiện tạo thêm đối tượng Mouse2 trên file main với các tham số truyền vào là tên = "Micky", tuổi = 10, cân nặng = 1Kg, quê hương="USA", năm sản xuất = "1998". Sau đó gọi đến hàm showInfo() để hiển thị thông tin.

Đa kế thừa

- Python cho phép đa thừa kế, điều đó có nghĩa là bạn có thể tạo ra một lớp mở rộng (extends) từ 2 hoặc nhiều lớp khác.
- Các lớp cha có thể có các thuộc tính giống nhau, hoặc các phương thức giống nhau. Lớp con sẽ ưu tiên thừa kế thuộc tính, phương thức của lớp đứng đầu tiên trong danh sách thừa kế.

❑ Cú pháp của đa kế thừa

```
from classparent1 import ClassParent1
from classparent2 import ClassParent2
from classparent3 import ClassParent3

# ClassChildren là lớp dẫn xuất hay lớp con
# ClassParent1, ClassParent2, ClassParent3 là lớp cha hay lớp được thừa kế

class ClassChildren(ClassParent1, ClassParent2, ClassParent3):

  # Code ...
```

Bài tập đa kế thừa

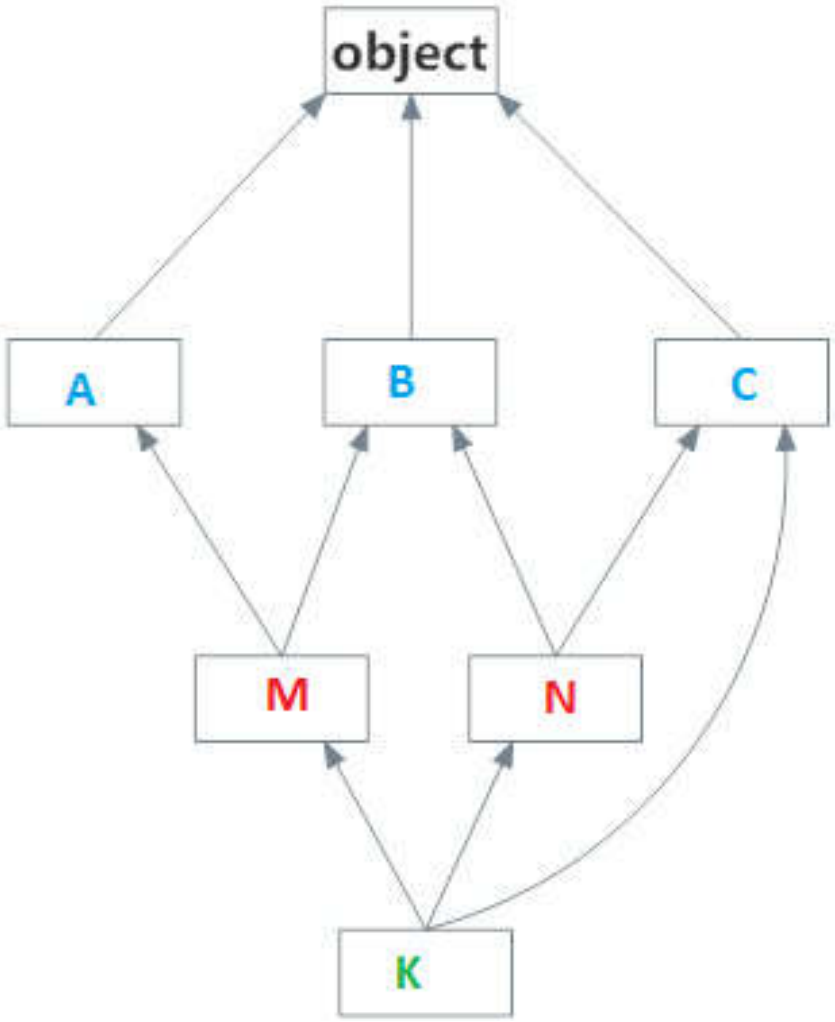
Bài 2:

- a) Tạo lớp Film (phim) bao gồm 2 thông tin tác giả (author) và loại phim (type)
- b) Tạo hàm showInfo() tại lớp Film. Hàm này để hiển thị thông tin bao gồm tên tác giả và loại phim.
- c) Tại lớp Mouse kế thừa lại 2 lớp là Animal và Film.
- d) Tại lớp main khởi tạo đối tượng Mouse2 với các thông tin như sau: tên = “Micky”, tuổi = 10, cân nặng = 1Kg, tác giả = “Walt Disney”, loại phim =“cartoon”. Gọi đến hàm showInfo để xem kết quả.

Phương thức mro()

- Phương thức **mro()** giúp bạn xem danh các lớp cha của một lớp nào đó

```
class A: pass
class B: pass
class C: pass

class M(A,B): pass
class N(B,C): pass

class K(N,M,C): pass

print(K.mro())
```


object
A
B
C
M
N
K

Đa hình

- **Polymorphism** - Tính đa hình: Thể hiện thông qua việc gửi các thông điệp (message). Việc gửi các thông điệp này có thể so sánh như việc gọi các hàm bên trong của một đối tượng. Các phương thức dùng trả lời cho một thông điệp sẽ tùy theo đối tượng mà thông điệp đó được gửi tới sẽ có phản ứng khác nhau.

- Tạo ra 2 lớp English và Vietnamese. Cả hai lớp này đều có phương thức greeting(). Cả hai tạo ra 2 lời chào khác nhau

```
# Tạo lớp English có hàm greting()
class English:

    def greeting(self):
        print ("Hello!")

 # Tạo lớp Vietnamese cũng có hàm greting()
class Vietnamese:

    def greeting(self):
        print ("Xin chao!")
```

- Tạo ra 2 đối tượng tương ứng từ 2 lớp trên và gọi hành động của 2 đối tượng này trong cùng một hàm (Hàm showIntro)

```
# Tạo hàm showIntro để hiển thị lời chào
def showIntro(language):
    language.greeting()

# Khởi tạo 2 đối tượng tương ứng với English và Vietnamese
en  = English()
vn = Vietnamese()

# Gọi đến 2 hàm showIntro của 2 đối tượng Enlish và Vietnamese
showIntro(en)
showIntro(vn)
```

THANKS FOR ATTENDING!