KẾ THỪA (INHERITANCE)

1. Đặt vấn đề:

VẤN ĐỀ

- Giả sử phần mềm của bạn cần quản lý thông tin của những đối tượng Sinh Viên, Giáo Viên, Nhân Viên của một trường đại học.
- Các đối tượng này có những thuộc tính chung ví dụ như tên, tuổi, ngày sinh, địa chỉ.

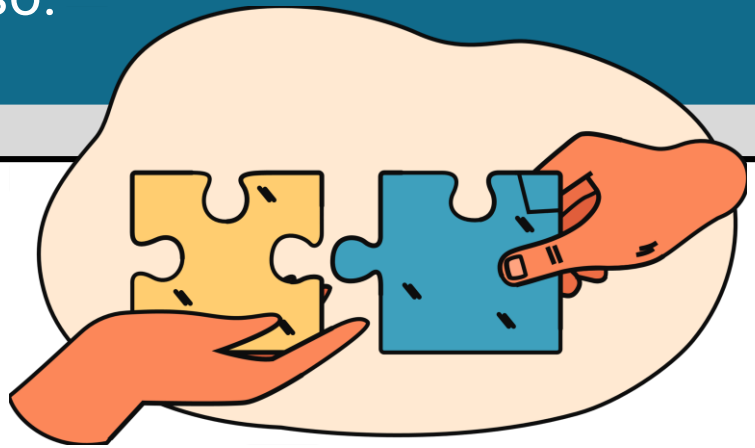
Vậy để có thể tránh được việc dư thừa code và tái sử dụng phần mềm thì hướng giải quyết là gì ?

1. Đặt vấn đề:

GIẢI PHÁP

Xây dựng một lớp cơ sở (base class) chứa các thuộc tính chung của 3 lớp này, sau đó cho các lớp này kế thừa từ lớp cơ sở.

Khi đó ta cần bổ sung các thuộc tính, phương thức của từng lớp dẫn xuất (derived class).



Lớp trong Python có thể mở rộng, tạo một lớp mới từ một lớp cũ mà vẫn bảo toàn được những đặc điểm của lớp cũ. Quá trình này gọi là kế thừa, kế thừa liên quan tới các khái niệm như lớp cha (base class hoặc super class), lớp con (derived class hoặc sub class).

2. Cú pháp kế thừa:

CÚ PHÁP:

```
class BaseClass:
class DerivedClass(BaseClass):
```

VÍ DỤ:

EXAMPLE

```
class Person:
    def __init__(self):
        print('Person constructor')

class Student(Person):
    def __init__(self):
        print('Student constructor')
```

3. Ví dụ về kế thừa:

Khi lớp con kế thừa lớp cha, lớp con sẽ có các thuộc tính, phương thức tương tự như lớp cha.

VÍ DỤ 1:

EXAMPLE

```
class Person:
    def greet(self):
        print('Hello 28tech')
class Student(Person):
    pass

if __name__ == '__main__':
    p = Person()
    p.greet()
    s = Student()
    s.greet()
```

OUTPUT

```
Hello 28tech
Hello 28tech
```

3. Ví dụ về kế thừa:

Khi lớp con kế thừa lớp cha, lớp con sẽ có các thuộc tính, phương thức tương tự như lớp cha.

VÍ DỤ 2:


```
class Person:
    count = 0
    def __init__(self):
        self.count += 1
    def getCount(self):
        return self.count
class Student(Person):
    def change(self):
        self.count += 1
if __name__ == '__main__':
    s = Student()
    print(s.getCount())
    s.change()
    print(s.getCount())
```

OUTPUT

```
1
2
```

4. Subclassing:

Thông thường lớp con sẽ có thêm những thuộc tính mới so với lớp cha, tuy nhiên để tránh dư thừa code, bạn có thể gọi hàm khởi tạo của lớp cha trong hàm khởi tạo của lớp con để khởi tạo các thuộc tính mà lớp con kế thừa từ lớp cha.


```
class Person:
    def __init__(self, name, birth):
        self.name = name
        self.birth = birth
class Student(Person):
    def __init__(self, name, birth, gpa):
        Person.__init__(self, name, birth)
        self.gpa = gpa
    def display(self):
        print(f'{self.name} {self.birth} {self.gpa:.2f}')
if __name__ == '__main__':
    s = Student('28tech', '01/01/2022', 2.5)
    s.display()
```

OUTPUT

```
28tech 01/01/2022 2.50
```

4. Subclassing:

Bạn cũng thể gọi hàm của lớp cha ở lớp con thông qua **từ khóa super()**

VÍ DỤ:


```
class Person:
    def __init__(self, name, birth):
        self.name = name
        self.birth = birth
class Student(Person):
    def __init__(self, name, birth, gpa):
        super().__init__(name, birth)
        self.gpa = gpa
    def display(self):
        print(f'{self.name} {self.birth} {self.gpa:.2f}')
if __name__ == '__main__':
    s = Student('28tech', '01/01/2022', 2.5)
    s.display()
```

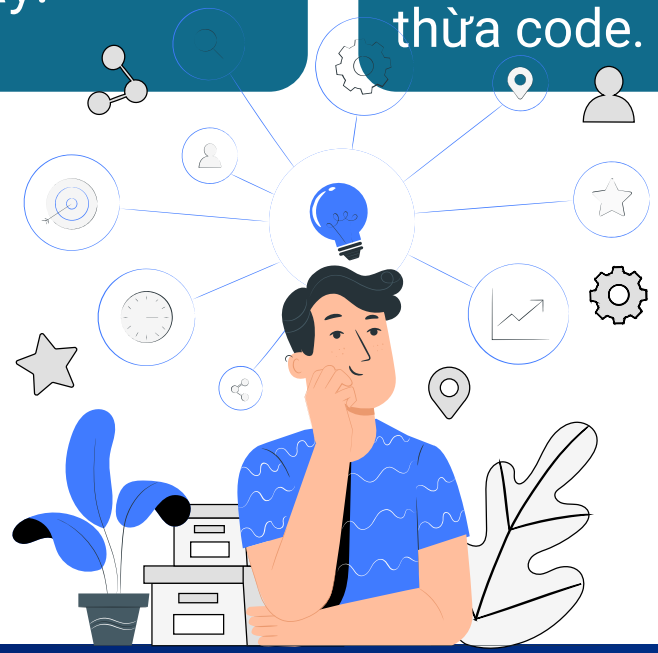
OUTPUT

```
28tech 01/01/2022 2.50
```

5. Ghi đè hàm:

Trong trường hợp lớp con và lớp cha có phương thức cùng tên, khi đó bạn đang ghi đè phương thức này.

Thông thường phương thức này ở lớp con sẽ được cài đặt một cách chi tiết hơn và có thể gọi phương thức tương ứng ở lớp cha để tránh dư thừa code.



5. Ghi đè hàm:

Lớp cha:

```
class Person:
    def __init__(self, name, birth):
        self.name = name
        self.birth = birth
    def show(self):
        return f'{self.name} {self.birth}'
```

Lớp con:

```
class Student(Person):
    def __init__(self, name, birth, gpa):
        super().__init__(name, birth)
        self.gpa = gpa
    def show(self):
        return Person.show(self) + ' ' + f'{self.gpa:.2f}'
```

Ví dụ:

```
if __name__ == '__main__':
    t = Person('Nam', '22/12/2002')
    print(t.show())
    s = Student('28tech', '01/01/2022', 2.5)
    print(s.show())
```

OUTPUT

```
Nam 22/12/2002
28tech 01/01/2022 2.50
```

6. Các kiểu kế thừa:

Đa kế thừa - Multiple Inheritance:

❖ Một lớp có thể cùng kế thừa nhiều lớp khác nhau được gọi là đa kế thừa.

Ví dụ:

EXAMPLE

```
class A:
    def __init__(self, a):
        self.a = a
class B:
    def __init__(self, b):
        self.b = b
class C(A, B):
    def __init__(self, a, b):
        A.__init__(self, a)
        B.__init__(self, b)
    def show(self):
        print(self.a, self.b)
c = C(100, 200)
c.show()
```

6. Các kiểu kế thừa:

Kế thừa nhiều mức - Multilevel inheritance:

❖ Khi lớp con lại có một lớp con khác kế thừa nó ta có kế thừa nhiều mức.

Ví dụ:

EXAMPLE

```
class A:
    def __init__(self, a):
        self.a = a
class B(A):
    def __init__(self, b):
        self.b = b
class C(B):
    def __init__(self, a, b):
        A.__init__(self, a)
        B.__init__(self, b)
    def show(self):
        print(self.a, self.b)
c = C(100, 200)
c.show()
```