파이썬 스터디

class inheritance

1. 상속(inheritance)이란?

클래스 상속은 부모 클래스(Parent Class, Super class)의 속성과 메소드를 자식 클래스(Child class, Sub class)에 물려주는 것으로, 자식 클래스는 부모 클래스의 속성과 메소드를 받는다.

- 사용법

```
class Parent_Class:
  <내용>

class Child_Class(Parent_Class):
  <내용>
```

1-1. 상속 사용해보기

In [1]:

```
class Country:
    """Parent Class/Super Class"""

    name = '국가명'
    population = '인구'
    capital = '수도'

    def show(self):
        print('국가 클래스의 메소드입니다.')
```

In [3]:

```
class Korea(Country):
    """Child class/Sub Class"""

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

    def show_name(self):
        print('국가 이름은 : ', self.name)
```

In [4]:

```
a = Korea('대한민국')
a.show()
```

국가 클래스의 메소드입니다.

In [5]:

```
a.show_name()
```

국가 이름은 : 대한민국

In [6]:

```
a.capital
```

Out[6]:

'수도'

In [7]:

```
a.name
```

Out[7]:

'대한민국'

3. 메소드 오버라이딩(Method overriding)

- **메소드 오버라이딩**: 부모 클래스의 메소드를 자식 클래스에서 재정의해 사용하는 것.

In [12]:

```
class Korea2(Country):
    '''Child Class/Sub Class'''
    def __init__(self, name, population, capital):
        self.name = name
        self.population = population
        self.caltal = capital

    def show(self):
        super().show()
        print(f"""
        국가의 이름은 {self.name}입니다.
        국가의 인구는 {self.population:,}입니다.
        국가의 수도는 {self.capital}입니다.
        """)
```

In [13]:

```
a = Korea2('대한민국', 50000000, '서울')
a.show()
```

국가 클래스의 메소드입니다.

국가의 이름은 대한민국입니다.
국가의 인구는 50,000,000입니다.
국가의 수도는 수도입니다.

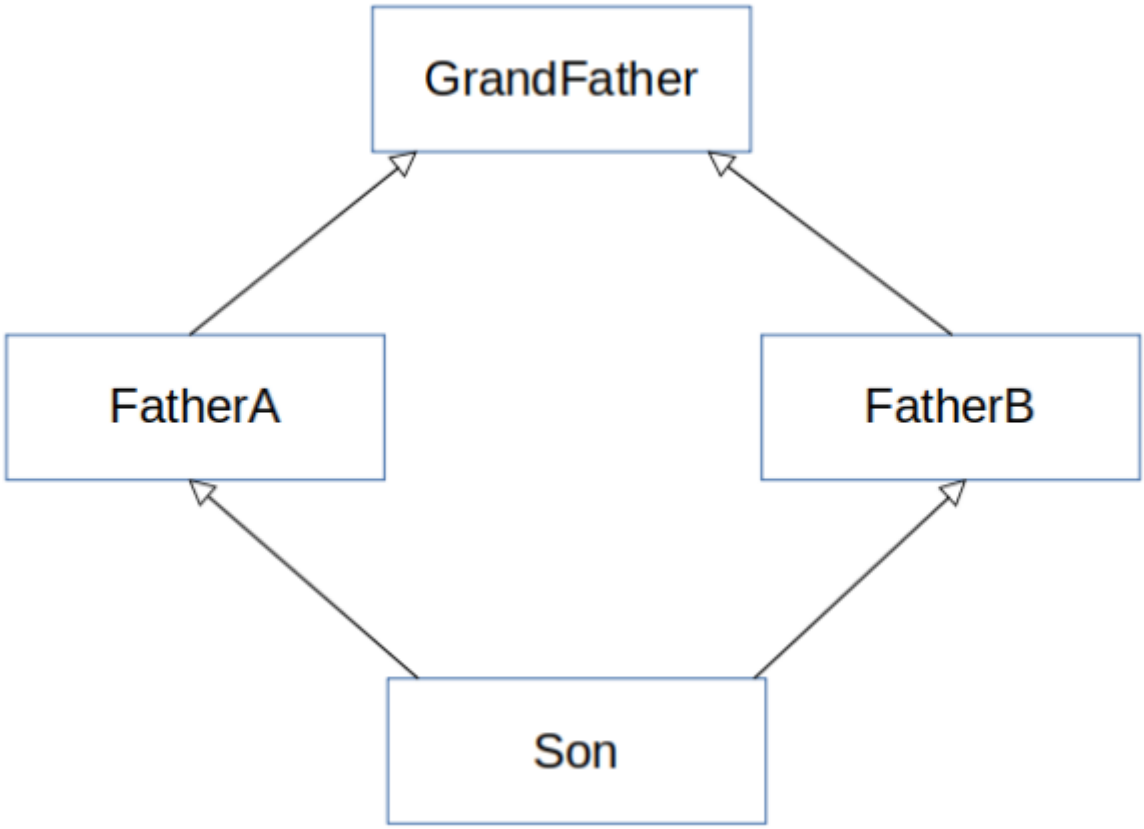
3. 다중상속

다중 상속은 두 개 이상 클래스를 동시에 상속받는 것이다. 파이썬과 C++은 다중상속이 가능하나 C#과 Java는 불가능하다.

다중상속 문제점

다중상속을 사용하는 이유로 편리함을 꼽는다. 그러나 다중 상속은 여러가지 문제를 가지고 있다. 다중상속은 사용하다보면 코드의 흐림이 모호해지며, 충돌·오류를 야기할 수 있다.

- 다이아몬드 문제



위 그림을 보면 FatherA와 FatherB에 같은 이름의 메소드가 존재한다면 어떤 메소드가 실행될까? son의 입장에서는 어떤 부모의 메서드를 상속받는지 예측하기 어렵다. 이러한 모호함은 충돌이나 오류를 야기할 수 있다. 이를 다이아몬드 문제라고 부른다.
위 문제의 답을 하자면, 왼쪽에 쓰여진 클래스의 메소드가 실행된다.

In [19]:

```
class Person:
    """Parent Class/Super Class"""
    def greeting(self):
        print('안녕하세요. Person 클래스입니다.')

class University:
    """Parent Class/Super Class"""
    def greeting(self):
        print('안녕하세요. University 클래스입니다')

class Undergraduate(Person, University):

    def greeting(self):
        super().greeting()
```

In [20]:

```
b = Undergraduate()
b.greeting()
```

안녕하세요. Person 클래스입니다.

Overloading

오버로딩은 메소드를 중복 정의하는것으로 같은 이름의 메소드를 사용하는 것. 파이썬에서는 오버로딩을 지원하지 않고, 같은 이르므이 메소드를 사용하면 늦게 정의한 메소드를 덮어쓰기한다.
메소드 명은 같지만 매개변수가 다르거나, 리턴 타입이 다르면 오버로딩

In [24]:

```
class A:
    def func(slef, a):
        return 'Hello'
    def func(self):
        return 'good morning'
```

In [26]:

```
o1 = A()
o1.func()
```

Out[26]:

'good morning'

1. 연산자 오버로딩

기존 약속되어 있는 __add__ 메소드를 재정의해서 해당 클래스에 객체 간 덧셈 연삼을 가능하게 함
→ 특수한 메소드 만들 때 사용
→ 인스턴스의 사칙연산을 만들 때
연산자 오버로딩을 하지 않으면 인스턴스간의 연산이 되지 않는다.

In [28]:

```
class A:
    def __init__(self, i):
        self.i = i

    def __add__(self, other):
        return self.i + other
```

In [29]:

```
n = A(20)
n + 1
```

Out[29]:

21

정리하기

- 오버라이딩은 상속된 후 메소드가 부모 메소드의 매개변수, 리턴 타입도 같아야 오버라이딩
- 오버로딩은 상속됐더라도 부모 메소드와 매개변수가 다르거나, 리턴 타입이 다르면 오버로딩

In [ ]:

```

```