클래스 상속

Python을 활용한 자료구조 이해하기

클래스 상속

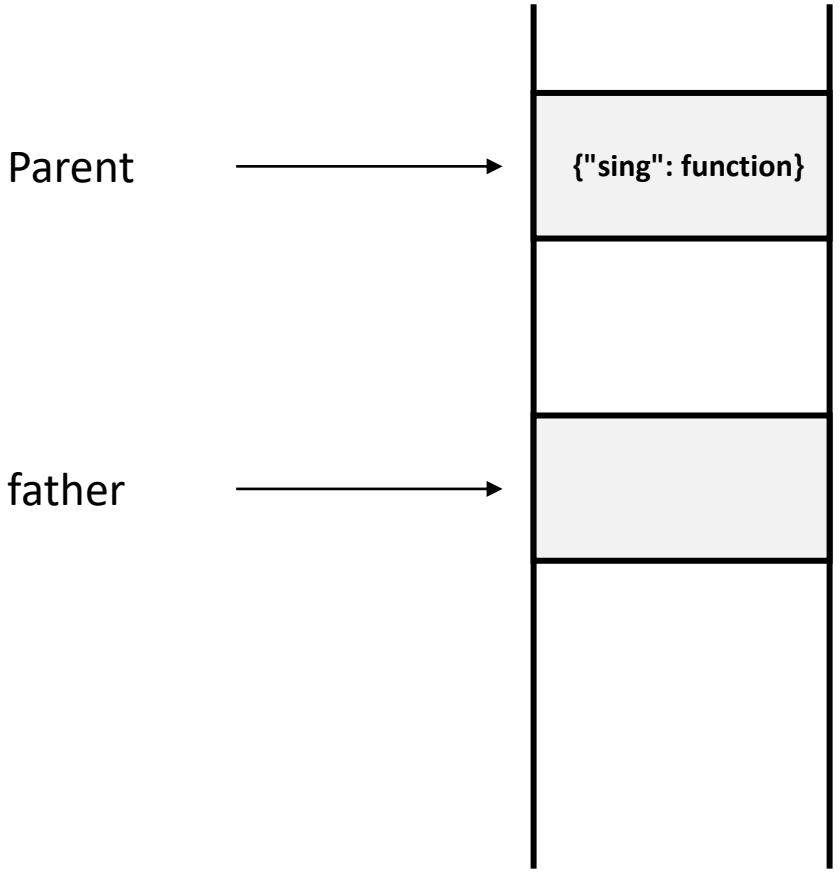
- 상속 (inheritance)는 기존 클래스의 작동 방식을 변경함으로써 새로운 클래스를 만드는 메커니즘
- 자식이 부모님으로부터 재산 등을 상속받는 것처럼 다른 클래스에 구현된 메서드나 속성값들을 상속받는 클래스에서 그대로 사용할 수 있음

노래 잘 부르는 부모 클래스

- Parent 클래스는 sing 메서드를 가짐

```
class Parent:
    def sing(self):
        print("sing a song")

father = Parent()
father.sing()
```



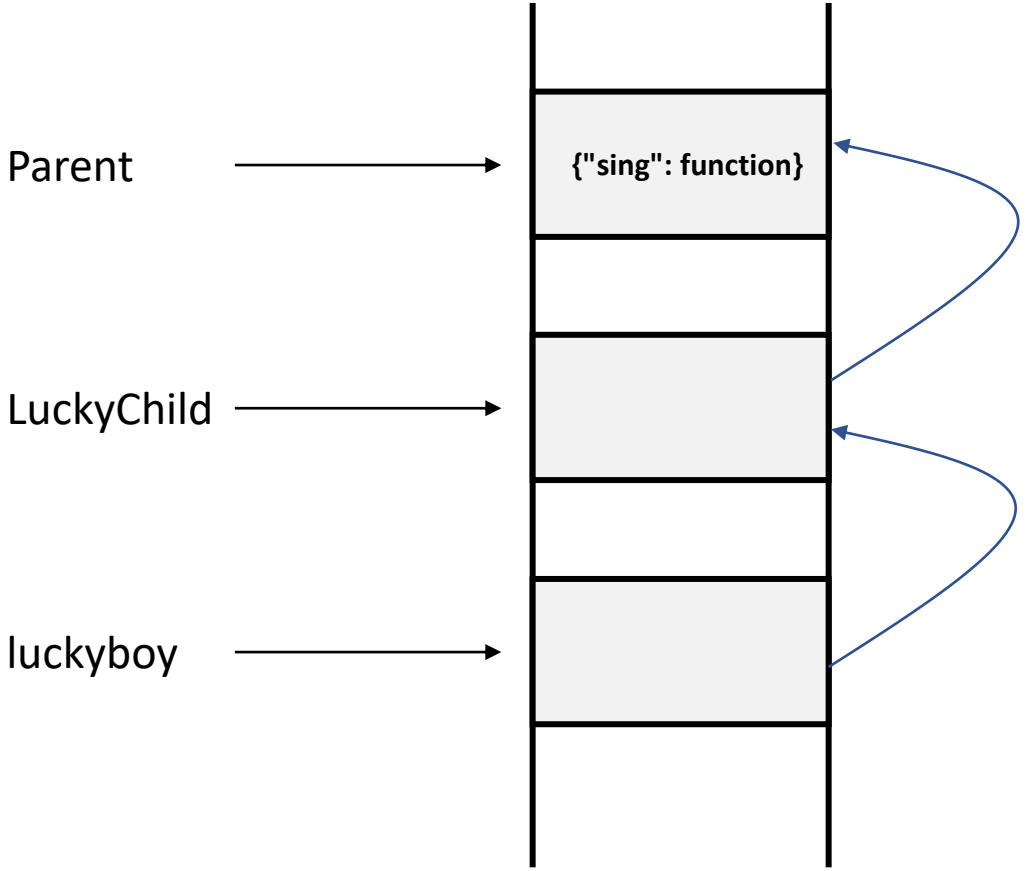
운이 좋은 자식 클래스

- 부모로부터 능력을 물려 받음

```
class Parent:
    def sing(self):
        print("sing a song")

class LuckyChild(Parent):
    pass

luckyboy = LuckyChild()
luckyboy.sing()
```



운이 없는 자식 클래스

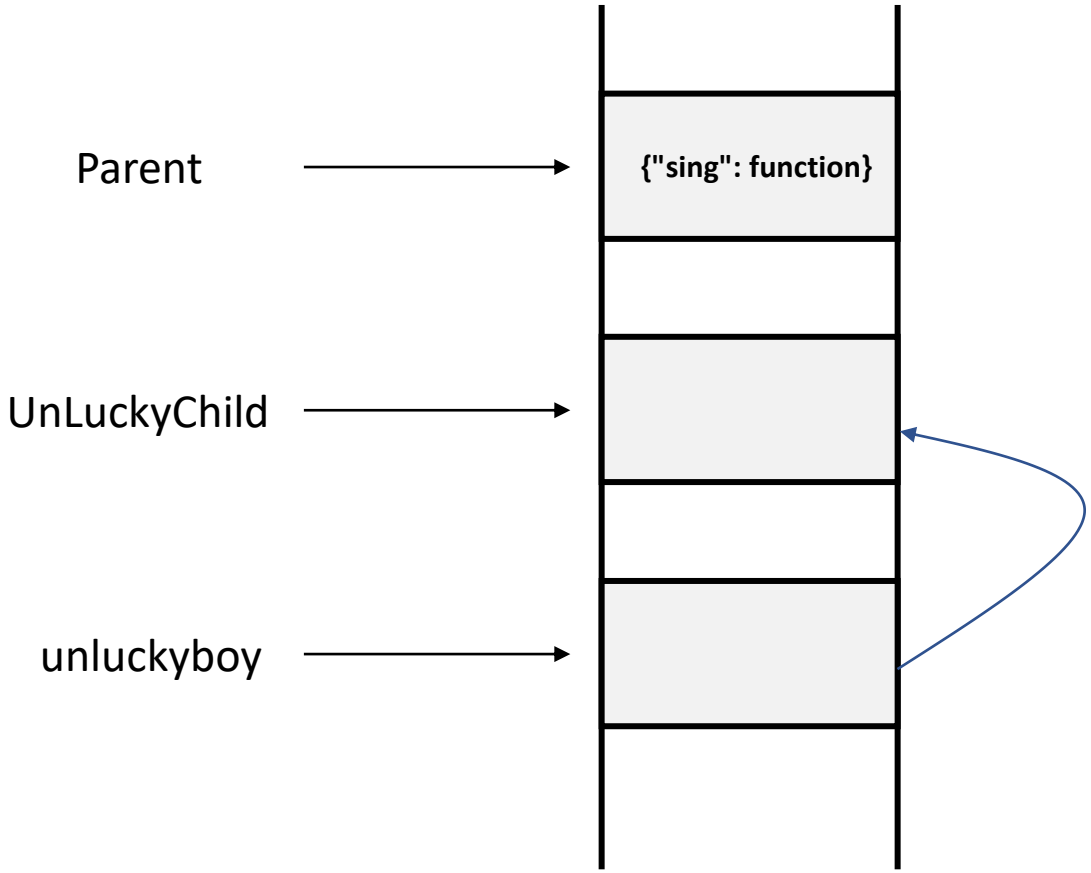
- 부모로부터 능력을 물려 받지 못한 자식 클래스

```
class Parent:
    def sing(self):
        print("sing a song")

class UnLuckyChild:
    pass

unluckyboy = unLuckyChild()
unluckyboy.sing()
```

실행결과
```
 unluckyboy.sing()
AttributeError: 'UnLuckyChild' object has no attribute 'sing'
```



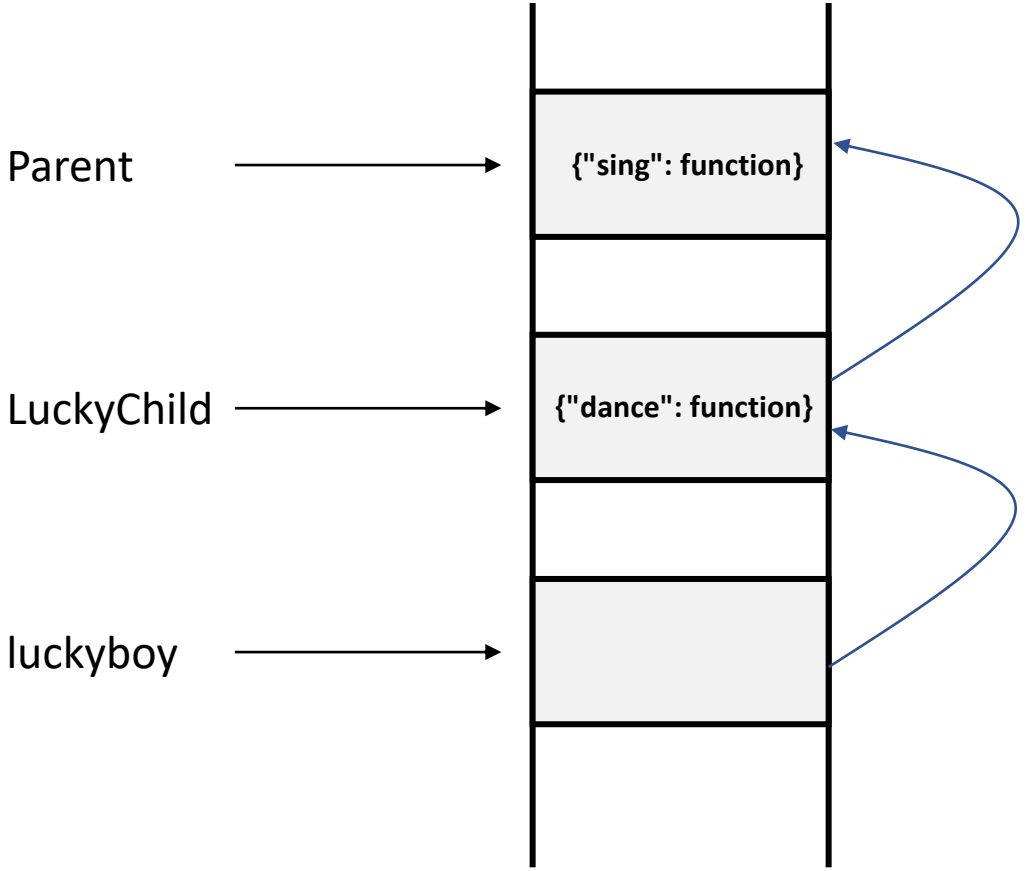
바람직한 상속 케이스

- 노래 능력은 부모로부터 물려받고 춤은 노력해서…

```
class Parent:
    def sing(self):
        print("sing a song")

class LuckyChild(Parent):
    def dance(self):
        print("shuffle dance")

luckyboy = LuckyChild()
luckyboy.sing()
luckyboy.dance()
```



인스턴스 속성 참조 (1/3)

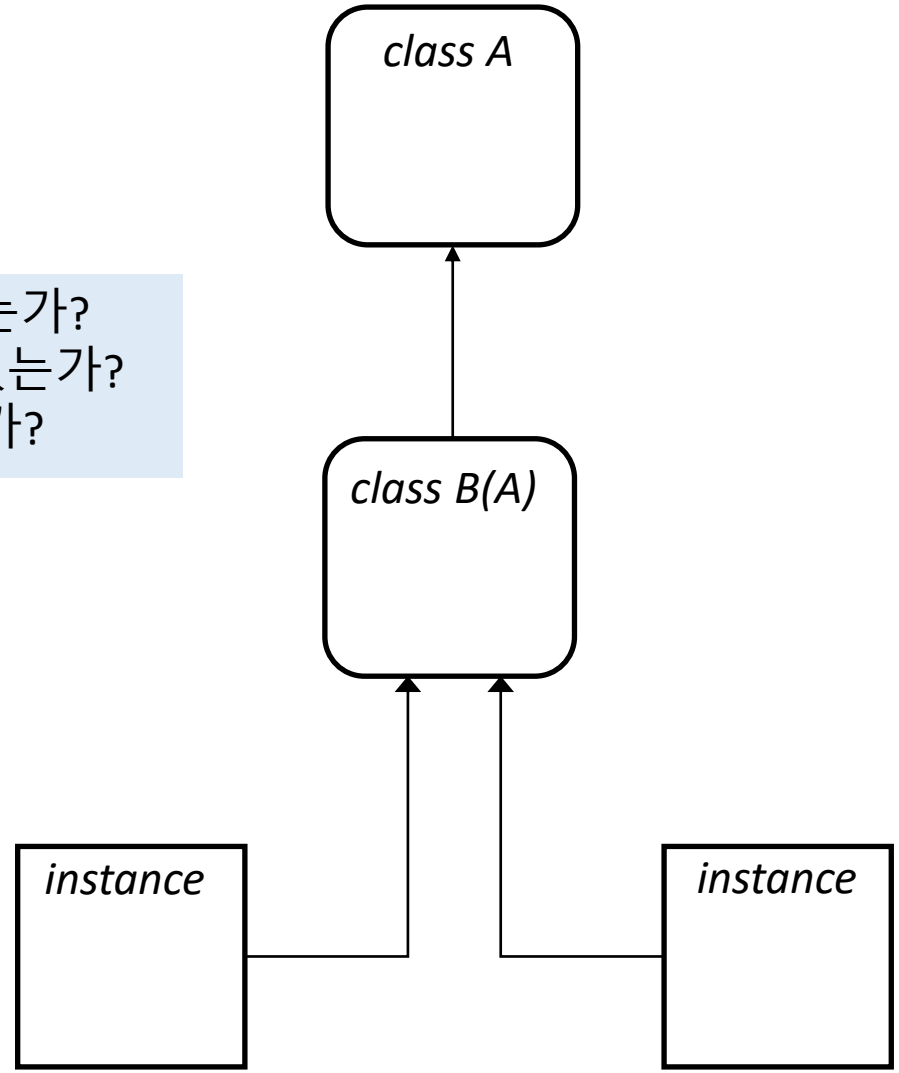
- 속성 참조 (property/method)
  - object.attribute

```
class Parent:
    def sing(self):
        print("sing a song")

class LuckyChild(Parent):
    def dance(self):
        print("shuffle dance")

luckyboy = LuckyChild()
luckyboy.sing()
```

1) instance에 sing()이 있는가?
2) LuckyChild에 sing()이 있는가?
3) Parent에 sing()이 있는가?



인스턴스 속성 참조 (2/3)

- 먼저 탐색된 속성/메서드를 우선 호출

```
class Parent:
    def sing(self):
        print("sing a song")

class LuckyChild(Parent):
    def sing(self):
        print("song?")

luckyboy = LuckyChild()
luckyboy.sing()
```

```
song?
```

인스턴스 속성 참조 (3/3)

- 명시적으로 부모 클래스를 호출하는 코드를 추가

```
class Parent:
    def sing(self):
        print("sing a song")

class LuckyChild(Parent):
    def sing(self):
        Parent.sing(self)
        print("song?")

luckyboy = LuckyChild()
luckyboy.sing()
```

(O)

```
class Parent:
    def sing(self):
        print("sing a song")

class LuckyChild(Parent):
    def sing(self):
        self.sing(self)
        print("song?")

luckyboy = LuckyChild()
luckyboy.sing()
```

(X)

```
class Parent:
    def sing(self):
        print("sing a song")

class LuckyChild(Parent):
    def sing(self):
        super().sing()
        print("song?")

luckyboy = LuckyChild()
luckyboy.sing()
```

(O)

스타크래프로 보는 상속 (1/2)

- 비슷한 보병 유닛은 비슷한 속성을 갖고 있음

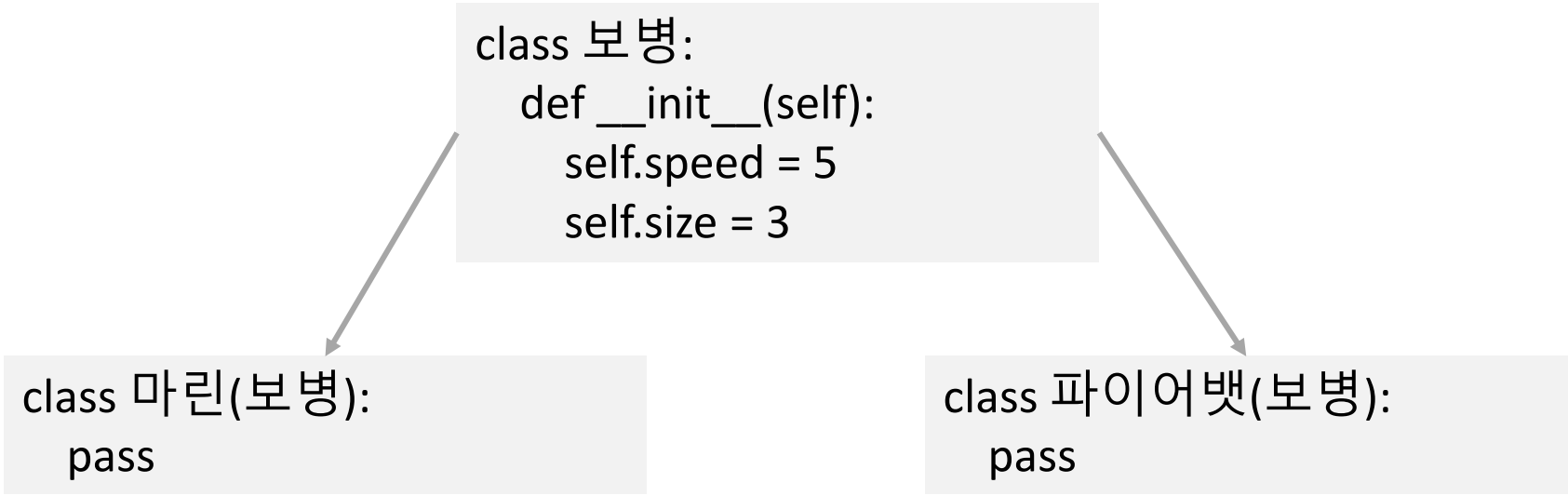
```
class 마린:
    def __init__(self):
        self.speed = 5
        self.size = 3
```

```
class 파이어뱃:
    def __init__(self):
        self.speed = 5
        self.size = 3
```

보병 유닛을 강화하고 싶어서 speed를 늘리고 싶다면?
-> 보병 유닛 클래스 숫자 만큼의 수정이 필요함

스타크래프로 보는 상속 (2/2)

- 공통된 속성을 부모 클래스로 정의



스타크래프로 보는 참조 순서 (1/2)

- 마린과 파이어뱃은 차이점이 존재함
  - 공통된 특성과 차이점을 어떻게 표현할 수 있을까?

```
class 마린:
    def __init__(self):
        self.attack = 3
        self.speed = 5
        self.size = 3
```

```
class 파이어뱃:
    def __init__(self):
        self.attack = 5
        self.speed = 5
        self.size = 3
```

스타크래프로 보는 참조 순서 (2/2)

- 부모 클래스를 호출하고 차이점을 추가 기술