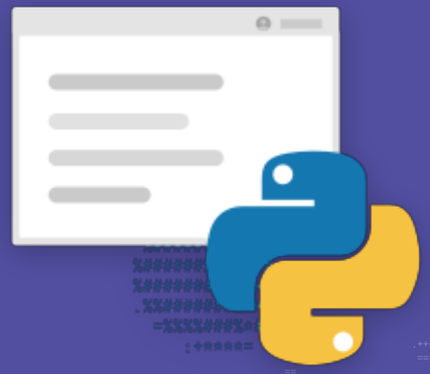
/* elice */

본격! 프로그래밍

클래스 더 알아보기



김건우 선생님

목차

커리큘럼

3 **클래스 더 알아보기**

여러 클래스의 관계를 정의하는 상속 개념에 대해 배웁니다.

데이터의 관계를 정의하며 프로그램의 큰 그림을 그려 봅니다.

4 **모듈과 패키지**

다른 사람이 만든 프로그램을 내 프로그램에서 사용할 수 있게 만들어 주는 모듈과 패키지에 대해 배웁니다.

파이썬의 다양한 오픈 소스 패키지를 체험해 봅니다.

클래스의 상속

왜 상속이 필요한가요?

여러 클래스가 **비슷한 속성과 메소드**를 공유해야 할 때

왜 상속이 필요한가요?

서로 다른 클래스 간의 **계층 구조**가 확실할 때

페이스북 게시물

게시물

- 글만 있는 게시물
- 사진을 포함한 게시물
- 동영상을 포함한 게시물
- 링크를 포함한 게시물

게시물

```
class Post:
    def __init__(self, content):
        self.content = content
```

이미지가 있는 게시물

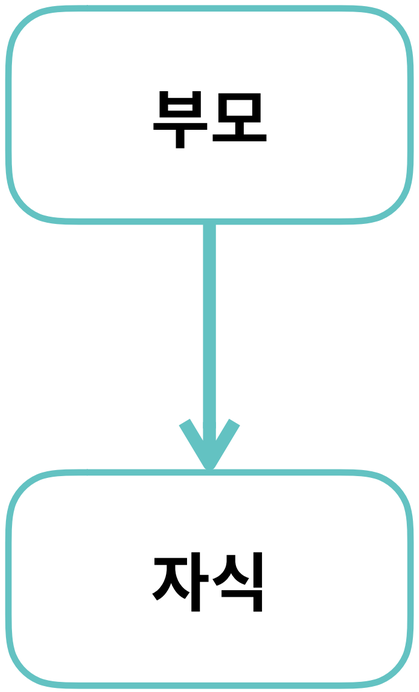
```
class ImagePost:
    def __init__(self, content, images):
        self.content = author
        self.images = images
```

이미지가 있는 게시물

```
class ImagePost:
    def __init__(self, content, images):
        ...

    def num_images(self):
        return len(self.images)
```

클래스의 상속



부모의 속성, 메소드

부모 + 자식의 속성, 메소드

청출어람의 법칙

부모 클래스보다 **자식** 클래스가
더 많은 데이터와 기능을 갖고 있다!

상속 따라하기

클래스 선언

```
class ImagePost(Post):
    ...
```

생성자

```
class ImagePost(Post):
    def __init__(self, content, images):
        super().__init__(content)
        self.images = images
```

super

```
class ImagePost(Post):
    def __init__(self, content, images):
        super().__init__(content)
        self.images = images
```

부모 클래스에 접근할 때

부모 인스턴스 동시 생성

```
class ImagePost(Post):
    def __init__(self, content, images):
        super().__init__(content)
        self.images = images
```

자식을 생성하면 연결되는 부모 인스턴스도 생성됨

속성 상속

```
class Post:
    def __init__(self, content):
        self.content = content
        self.likers = []
```

속성 상속

```
class ImagePost(Post):
    ...

my_post = ImagePost(alice, "#강남맛집")
print(my_post.likers)    # []
```

메소드 상속

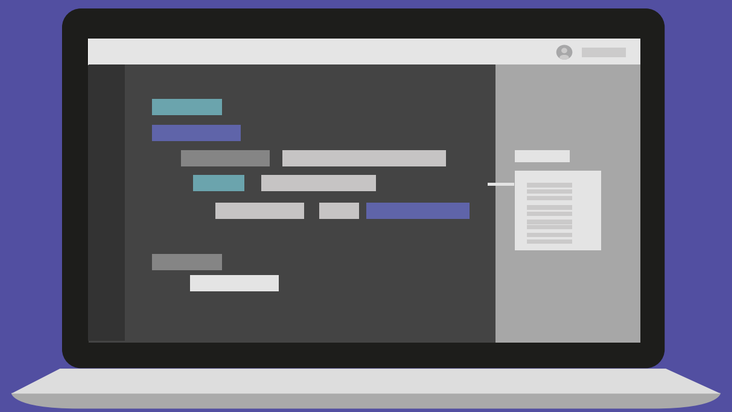
```
class Post:
    def like(self, user):
        self.likers.append(user)
```

메소드 상속

```
class ImagePost(Post):
    ...

my_post = ImagePost(alice, "#강남맛집")
my_post.like(bob)
```

[실습 1] 링크가 있는 게시물



좋아요, 슬퍼요

```
class Like:
    def __init__(self, post, user):
        self.post = post
        self.user = user
```

좋아요, 슬퍼요

```
class Sad:
    def __init__(self, post, user):
        self.post = post
        self.user = user
```

추상적인 부모 클래스

```
class Reaction:
    def __init__(self, type, post, user):
        self.type = type
        self.post = post
        self.user = user
```

추상적인 부모 클래스

```
class Like:
    def __init__(self, post, user):
        super().__init__("LIKE", post, user)
```

추상적인 부모 클래스

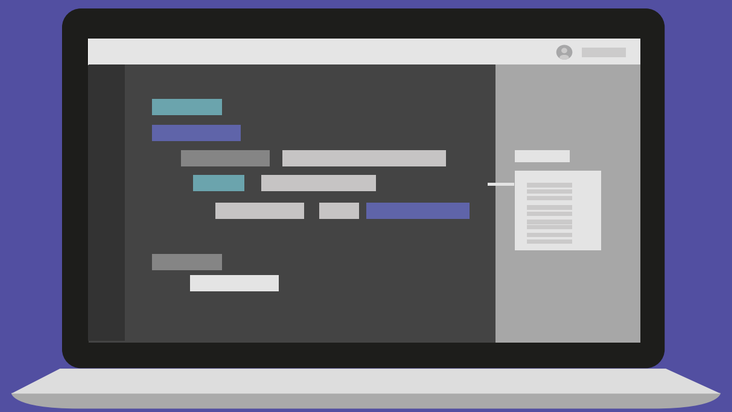
```
class Sad:
    def __init__(self, post, user):
        super().__init__("SAD", post, user)
```

추상적인 부모 클래스

```
// 이렇게 쓰진 않는다!
reaction = Reaction("vasdjkl", post, me)

// 반드시 구체적인 자식 클래스로 쓴다.
like = Like(post, me)
```

[실습 2] 게시물에 반응하기



오버라이딩

클래스 설계 상황

모든 게시물에는 댓글을 달 수 있지만,

보호 설정을 한 게시물에는 댓글을 달 수 없다.

오버라이딩

```
class Post:
    def comment(self, user, content):
        self.comments.append(Comment(user, content))

class ProtectedPost(Post):
    def comment(self, user, content):
        print("Can't comment on protected posts.")
```

오버라이딩

```
class Post:
    def comment(self, user, content):
        self.comments.append(Comment(user, content))

class ProtectedPost(Post):
    def comment(self, user, content):
        print("Can't comment on protected posts.")
```

[실습 3] 게시물 나만 보기