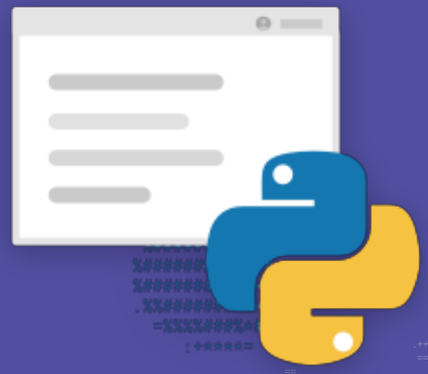
/* elice */

본격! 프로그래밍

프로그래밍 세상의 설계도, 클래스 입문



김건우 선생님

목차

커리큘럼

1

코드의 기본, 함수

코드의 불필요한 반복을 줄여주고, 더 이해하기 쉬운 코드를 만들어 주는 함수에 대해 배워봅니다.

2

프로그래밍 세상의 설계도, 클래스 입문

보다 논리적인 프로그램을 설계할 수 있게 도와주는 클래스 개념에 대해 배우고, 직접 클래스를 설계해 봅니다.

이것이 클래스

클래스란?

나타내고자 하는 개념의 **설계도**

페이스북 게시물

저장해야 하는 데이터	할 수 있는 조작
작성자	리액션 (좋아요, 슬퍼요, ...)
내용	댓글 달기
작성한 날짜/시간	내용 수정하기
이미지 (없을 수도 있음)	공유하기
링크 (없을 수도 있음)	사진 추가하기

클래스와 인스턴스

클래스

어떤 데이터가 있는지,

어떤 조작을 할 수 있는지,

어떤 제약조건들이 있는지

명시한 추상적인 설계도

인스턴스

그 클래스로 만든

실제 예시

클래스와 인스턴스

게시물 클래스

게시물 하나에 최대 20장의 사진

좋아요, 슬퍼요, 최고에요

댓글을 달 수 있음

작성자가 공유 여부를 설정 가능

게시물 인스턴스

elice의 7월 17일 게시물

최XX님의 1월 3일 게시물

리X왕 김X뷰의 7월 2일 게시물

...

클래스 선언

```
class Post:
    author = None
    comments = []
    likes = 0
    content = "What are you doing?"
```

속성

클래스 선언

```
class Post:
    ...
    def like(self, user):
        self.likes += 1
        user.liked_posts.append(self)
```

메소드

클래스 하나씩 따라하기

생성자

모든 클래스의 가장 기본이 되는 메소드

인스턴스가 처음 만들어질 때 어떻게 세팅할 것인지 결정

생성자

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

생성자의 매개변수 vs. 클래스의 속성

계좌개설 신청서 (고객용)

이름	**김코딩**
추천직원	**엘리스**

계좌개설 확인서 (내부용)

고객명	**김코딩**
추천직원번호	**2435**

생성자의 매개변수 vs. 클래스의 속성

계좌개설 신청서 (고객용)

이름	김코딩
추천직원	엘리스

생성자의 매개변수

(인스턴스 생성 시 입력)

계좌개설 확인서 (내부용)

고객명	김코딩
추천직원번호	2435

클래스의 속성

(실제로 데이터가 저장되는 이름)

생성자

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

self

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

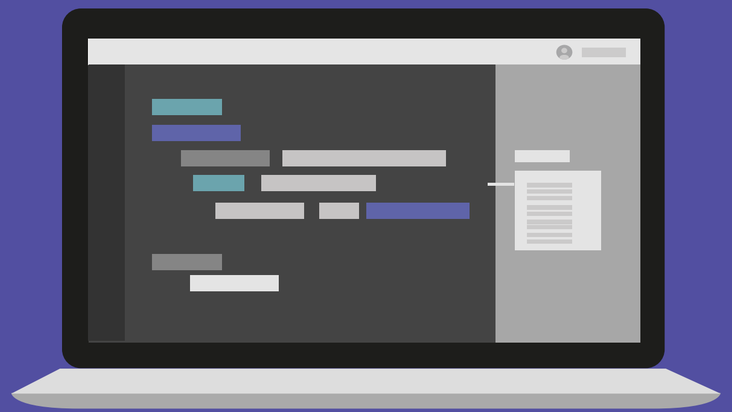
클래스 내부의 속성/메소드에 접근할 때

생성자

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

my_post = Post("elice", "I love coding!")
```

[실습 1] 첫 게시물



속성을 만들 때 주의할 점

```
class Post:
    def __init__(self, author, content):
        self.likes = 0
        self.liked_users = []
```

같은 내용을 나타내는 속성이 2개

속성을 만들 때 주의할 점

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

my_post = Post("elice", "I love coding!")
my_post.likes += 1
```

메소드 만들기

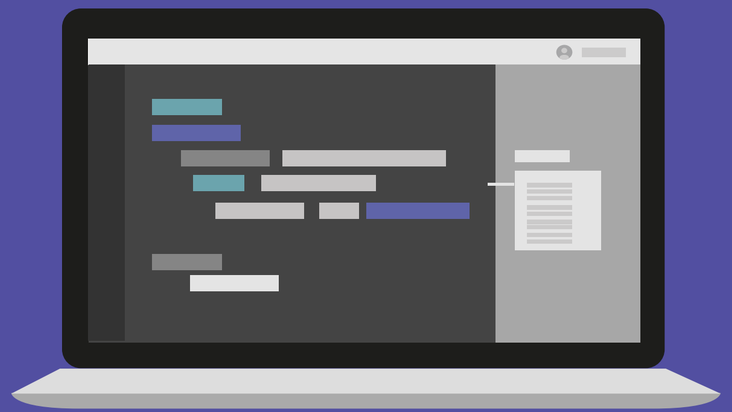
```
class Post:
    def like(self, user):
        self.liked_users.append(user)
```

메소드 만들기

```
class Post:
    def num_likes(self):
        return len(self.liked_users)
```

메소드를 속성처럼 사용

[실습 2] 좋아요 누르기



클래스 다듬기

원하지 않는 값 배제하기

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

my_post = Post("elice", 1457)
my_post.like(["Hello", "World"])
```

잘못된 값

원하지 않는 값 배제하기

```
class User:
    def __init__(self, year_of_birth):
        if type(year_of_birth) is not int:
            return
```

원하지 않는 값 배제하기

```
class Post:
    def __init__(self, author, content):
        if not isinstance(author, User):
            return
        if type(content) is not str:
            return
```

원하지 않는 값 배제하기

```
class User:
    def __init__(self, year_of_birth):
        if year_of_birth > 2005:
            raise Exception("Too young")
```

[실습 3] 가입하기