

클래스를 쓰는 이유는 기존의 자료형들로 간단히 표현되지 않는 것들을 표현하기 위해서입니다.

인스턴스 변수

SNS의 이용자들을 표현하기 위해서는 **User** 라는 자료형을 만들었는데, 각 이용자는 '이름', '나이', '이메일', '비밀번호' 등의 값을 보관해야 합니다.

다행히 각 인스턴스는 여러가지 **인스턴스 변수**를 보관할 수 있습니다.

```
class User:
    pass

# 첫번째 유저 값 설정
user1 = User()
user1.name = "Bill Gates"
user1.age = 60
user1.email = "bill@microsoft.com"
user1.password = "gates1234"
```

위의 코드에서는 **user1** 의 인스턴스 변수 **name**, **age**, **email**, 그리고 **password** 를 정의하였습니다. 이 변수들을 '인스턴스 변수'라고 부르는 이유는 각 변수가 **User** 클래스 전체에 해당되는 것이 아니라 **User** 클래스의 한 인스턴스인 **user1** 에 해당되기 때문입니다.

저장된 인스턴스 변수를 불러오려면 그냥 아래처럼 쓰면 됩니다:

```
print(user1.name, user1.age, user1.email, user1.password)
```

```
Bill Gates 60 bill@microsoft.com gates1234
```

정의되지 않은 인스턴스 변수

정의되지 않은 인스턴스 변수를 불러오면 에러가 나옵니다.

```

class User:
    pass

# 첫번째 유저 값 설정
user1 = User()
user1.name = "Bill Gates"
user1.age = 60
user1.email = "bill@microsoft.com"
user1.password = "gates1234"

# 두번째 유저 값 설명
user2 = User()

print(user2.name, user2.age, user2.email, user2.password)

```

```

Traceback (most recent call last):
  File "e.py", line 14, in <module>
    print(user2.name, user2.age, user2.email, user2.password)
AttributeError: 'User' object has no attribute 'name'

```

따라서 모든 인스턴스 변수는 처음부터 초기값을 설정해주는 것이 중요합니다!

```

class User:
    pass

# 첫번째 유저 값 설정
user1 = User()
user1.name = "Bill Gates"
user1.age = 60
user1.email = "bill@microsoft.com"
user1.password = "gates1234"

# 두번째 유저 값 설명
user2 = User()
user2.name = "Mark Zuckerberg"
user2.age = 32
user2.email = "mark@facebook.com"
user2.password = "zuck123"

print(user1.name, user1.age, user1.email, user1.password)
print(user2.name, user2.age, user2.email, user2.password)

Bill Gates 60 bill@microsoft.com gates1234
Mark Zuckerberg 32 mark@facebook.com zuck123

```

메소드

클래스는 변수뿐만 아니라 함수도 보관할 수 있습니다. 클래스에 정의된 함수는 **메소드**라고 부릅니다. 이용자에 대한 간단한 정보를 출력하는 메소드를 써봅시다.

```
class User:
    def introduce(some_user):
        print("%s is %d years old." % (some_user.name, some_user.age))

# 첫번째 유저 값 설정
user1 = User()
user1.name = "Bill Gates"
user1.age = 60
user1.email = "bill@microsoft.com"
user1.password = "gates1234"

# 두번째 유저 값 설정
user2 = User()
user2.name = "Mark Zuckerberg"
user2.age = 32
user2.email = "mark@facebook.com"
user2.password = "zuck123"

User.introduce(user1)
User.introduce(user2)
```

```
Bill Gates is 60 years old.
Mark Zuckerberg is 32 years old.
```

이렇게 **User** (클래스 이름), **.** (마침표), **introduce** (메소드 이름)을 쓰면 메소드를 호출할 수 있습니다. 여기에 파라미터로 인스턴스 **user1**을 넘겨주면 **user1.name**과 **user1.age**가 불러오고 문자열 포매팅에 의해 **"Bill Gates is 60 years old."**가 출력되는거죠. 마찬가지로 파라미터로 **user2**를 넘겨주면 **"Mark Zuckerberg is 32 years old"**가 출력됩니다.

Syntactic Sugar

그러나 대부분의 경우에 파라미터로 **user1**이나 **user2** 같은 **User** 클래스의 인스턴스를 넘겨줄 것이기 때문에, 파이썬에서는 아예 메소드 호출을 쉽게 쓰는 방법을 만들어 놓았습니다.

```
# 이 두 줄은 같음
User.introduce(user1)
user1.introduce()

# 이 두 줄은 같음
User.introduce(user2)
user2.introduce()
```

```
Bill Gates is 60 years old.
Bill Gates is 60 years old.
Mark Zuckerberg is 32 years old.
Mark Zuckerberg is 32 years old.
```

`user1.introduce()` 처럼 앞에 클래스 이름(`User`)이 아닌 인스턴스(`user1`)를 쓰면, 그 인스턴스(`user1`)가 `introduce` 메소드의 첫번째 파라미터로 넘어갑니다. 따라서 `user1.introduce` 는 `User.introduce(user1)` 과 똑같은 Syntactic Sugar인 셈이죠.

파라미터가 여러개인 경우를 봅시다. `introduce` 메소드가 두번째 파라미터로 정수 `n` 을 받고 자기 소개를 `n` 번 하도록 써보겠습니다.

```
class User:
    def introduce(some_user, n):
        for i in range(n):
            print("%s is %d years old." % (some_user.name, some_user.age))
```

이 경우에 Syntactic Sugar를 활용하여 쓴다면 이렇게 됩니다:

```
# 이 두 줄은 같음
User.introduce(user1, 3)
user1.introduce(3)

# 이 두 줄은 같음
User.introduce(user2, 2)
user2.introduce(2)
```

```
Bill Gates is 60 years old.
Bill Gates is 60 years old.
Bill Gates is 60 years old.
Bill Gates is 60 years old.
Bill Gates is 60 years old.
Bill Gates is 60 years old.
Mark Zuckerberg is 32 years old.
Mark Zuckerberg is 32 years old.
Mark Zuckerberg is 32 years old.
Mark Zuckerberg is 32 years old.
```


self


`user1.introduce()` 를 호출할때처럼 보통 첫번째 파라미터로 현재 주인공인 인스턴스를 넘겨줍니다. 그렇기 때문에 메소드의 첫번째 파라미터 이름은 **self** 라고 지어주는 것이 파이썬 커뮤니티의 약속입니다. **some_user** 라고 지어줘도 오류가 나지 않지만 코드의 일관성을 위해 약속을 꼭 지켜주세요!

```
class User:
    def introduce(self):
        print("%s is %d years old." % (self.name, self.age))
```



수업을 완료하셨으면 체크해주세요.

 수강생 Q&A 보기

 [\(/questions?](#)
질문하기

assignment_id=184&sort_by=popular)
[\(/questions/new?](#)

assignment_id=184&op1=%ED%94%84%EB%A1%9C%EA%B7%

< 이전 강의
메소드 (/assignments/153)

다음 강의
initialize > (/assignments/154)