



Data Analytics Based Python

SECT9. 클래스 (Class)

IT Competency Improvement Training
Kim Jin Soo



- ◆ 클래스 이해하기
- ◆ 클래스 정의 및 불러오기
- ◆ 클래스 초기화 함수 `__init__()` 재정의
- ◆ 클래스 변수와 인스턴스 변수
- ◆ 데이터 은닉과 이름 장식, Data Hiding & Name Mangling
- ◆ 객체 지향의 꽃 상속, Inheritance
- ◆ 다형성, Polymorphism

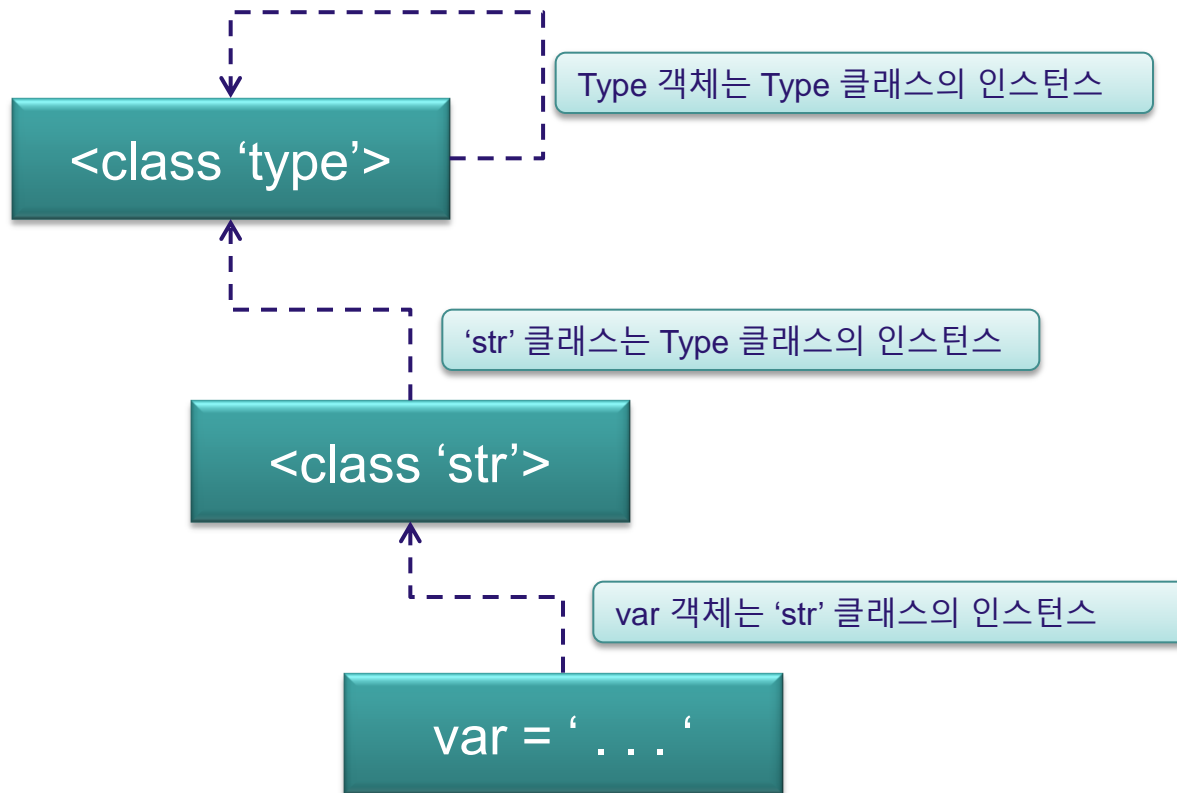


- ❖ 클래스는 객체지향언어에서 중요한 역할을 하고 있다.
 - 어떤 객체를 만드는 틀
 - SW를 설계할 때부터 오랜 기간 동안 SW개발의 명장들이 만들어 놓은 수많은 팁과 권고사항
 - 대부분의 개발자들이 따르는 개발 표준들을 구현하는 주체
- ❖ 모든 데이터형들은 클래스에 의해 생성된 것들이다.
- ❖ 타입과 클래스의 차이
 - 문자열형 변수 var의 타입은 str 라는 클래스
- ❖ 파이썬은 모든 것이 객체다
 - 클래스 역시 파이썬에서는 하나의 객체
- ❖ 클래스에서 생성된 객체를 인스턴스(Instance)라고 부른다.
 - 클래스의 객체를 생성하는 과정 → **인스턴스화**
 - 모든 인스턴스에는 타입이 존재, 이 타입은 클래스에 의해 정의 됨

클래스와 인스턴스 간의 관계



❖ Type, Class, Instance 의 관계



<인스턴스 표기법>



- ✓ B는 A의 인스턴스
- ✓ B가 A라는 명세서의 구현체라는 의미



❖ 클래스 안에는 크게 변수와 함수가 포함될 수 있다.

- 변수는 '속성' 이라고 부르고,
- 함수는 '메소드' 라고 부른다.

❖ 클래스 선언문

- class 예약어 후에 한 칸을 띄고 클래스명 입력
- 클래스명의 작성법은 낙타표기법(CamelCase)을 따른다.
 - 클래스명의 첫 음절인 대문자인 단어들의 조합형태로 작성
 - 변수나 함수의 이름은 소문자로만, 단어 사이는 언더바(_)로 구분

❖ 클래스 호출

- 클래스의 객체를 만드는 방법, 즉 인스턴스를 생성하는 방법은 함수를 호출하듯이 클래스명과 중괄호(())를 사용
- 매개변수 self
 - 클래스 내의 함수에 자동으로 주어지는 인자값
 - 본인의 객체를 인자 값으로 넘김으로써 함수 내에서 클래스의 속성 및 메소드에 접근할 수 있게 해준다.

Tip. 다른 객체 지향 언어에 익숙한 개발자



- ❖ 자바 기준으로 `self` 는 자바의 `this` 와 같다.
- ❖ 하지만 함수에 인자로 굳이 넣을 필요는 없다.
- ❖ 자바의 클래스라면 `name` 이라는 변수는 `read_book()` 함수에서 그냥 호출이 가능하다.
- ❖ 파이썬에서는 반드시 `read_book()` 선언시
 - `self` 를 첫 번째 매개변수로 선언해야 하며
 - 함수 내에서 클래스의 속성을 호출할 때도 반드시 `self.[속성명]` 형태로 사용해야 한다.
- ❖ '`self`' 라는 매개변수명은 사실 변경이 가능하나 묵시적으로 '`self`'를 사용하고 있으니 참고하기 바란다.

클래스 초기화 함수 `__init__()` 재정의



❖ `__init__()` 함수

- 인스턴스 생성시에 반드시 이름을 인자값으로 집어 넣게 하고 싶을 때
- 객체 인스턴스화를 위한 특수 함수
- 인자값 없이 객체를 생성하면 에러 발생
- 인스턴스 이름을 넣고 생성시, 메소드 호출 성공

```
# 클래스 정의
class BookReader:          # 클래스 BookReader 선언
    def __init__(self, name): # 초기화 함수 재정의
        self.name = name

    def read_book(self):     # 함수 선언
        print( self.name + ' is reading Book!!')

# 원래대로 인자값 없이 객체를 생성시, 에러가 발생
reader = BookReader()

-----
TypeError                                 Traceback (most recent call last)
<ipython-input-10-1533bb2d3088> in <module>()
----> 1 reader = BookReader()

TypeError: __init__() missing 1 required positional argument: 'name'

# 이름을 넣고 객체 생성
reader = BookReader('Daniel') # 객체 생성
reader.read_book()           # 메소드 호출

Daniel is reading Book!!
```

클래스 초기화 함수 `__init__()` 재정의



❖ `__init__()` 함수

- 인스턴스 생성시에 반드시 이름을 인자값으로 집어 넣게 하고 싶을 때
- 객체 인스턴스화를 위한 특수 함수

```
# 클래스 정의
class BookReader:          # 클래스 BookReader 선언
    def __init__(self, name): # 초기화 함수 재정의
        self.name = name

    def read_book(self):     # 함수 선언
        print( self.name + ' is reading Book!!')
```

```
# 원래대로 인자값 없이 객체를 생성시, 에러가 발생
reader = BookReader()
```

```
-----
TypeError                                 Traceback (most recent call last)
<ipython-input-10-1533bb2d3088> in <module>()
----> 1 reader = BookReader()

TypeError: __init__() missing 1 required positional argument: 'name'
```

```
# 이름을 넣고 객체 생성
reader = BookReader('Daniel') # 객체 생성
reader.read_book()            # 메소드 호출
```

```
Daniel is reading Book!!
```




❖ 변수의 선언 위치에 따라 달라지는 유효범위에 대해 알아보자

❖ 클래스 변수

- 클래스에 선언된 속성
- 클래스에 의해 생성된 모든 객체에서 같은 값을 조회할 때 사용이 가능

❖ 인스턴스 변수

- `__init__()` 함수 내에 선언된 변수
- 객체가 인스턴스화 될 때마다 새로운 값이 할당되며 서로 다른 객체 간에는 값을 공유할 수 없다.
- 객체간에 값을 공유할 필요가 있고, 값이 변경되지 않는 경우에는 클래스 변수를 활용할 수 있다.
- 변경이 되는 값을 클래스 변수로 활용하게 되면 본인의 의도와는 상관없이 다른 객체가 값을 수정하게 될 수도 있다.
- 즉, 객체 단위로 변경이 되는 변수는 반드시 인스턴스변수로 사용



❖ 데이터 은닉, Data Hiding

- 객체지향언어에서 캡슐화 Encapsulation 라고도 부른다.
- 데이터는 외부에 노출하지 말고 숨기자는 의도
- 파이썬에서는 이를 위한 강력한 도구를 제공하고 있지는 않다.
 - 자바에서는 다양한 접근 제한자(Private)를 가지고 있다.

✓ `dir()` : 클래스 내부에 들어있는 객체들을 확인하는 명령문

❖ 이름 장식, Name Mangling

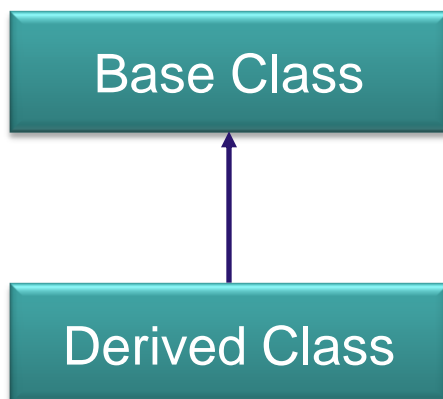
- 변수명 앞에 언더스코어 두 개(__)가 있는 것에 한하여 이름 변경 가능
- 변형된 규칙은 '`_[클래스명]_[변수명]`' 이다.
- 이렇게 변형된 변수는 기존 변수명으로는 값을 확인 할 수 없게 된다.



❖ 상속, Inheritance

- 부모 클래스가 자식 클래스에게 무언가를 물려 주는 것
- 파이썬에서의 부모클래스와 자식클래스 호칭
 - 부모 클래스 : 베이스 클래스, Base Class
 - 자식 클래스 : 파생 클래스, Derived Class
- 중복 코드를 최소화 함과 동시에 클래스간의 계층 관계를 형성함으로써 현실 세계와의 괴리를 줄이기 위해 이런 개념이 나왔다.

❖ 부모와 자식 클래스 관계 표기법



Base Class 는 부모 클래스, 자식 클래스들에게 물려줄 속성과 메소드를 정의한다.

Derived Class 는 자식 클래스로 부모의 속성, 메소드를 물려 받는다.



❖ 다형성, Polymorphism

- 부모 클래스와 동일한 이름의 메소드를 그대로 자식 클래스에서 구현하여 재정의의 Overriding 하는 것

❖ 행위 뿐만이 아니라 상태도 재정의 할 필요가 있다.

❖ 다형성의 구현을 위하여 부모 클래스에 있는 메소드를 재정의 하고 나면 부모 클래스의 메소드는 실행이 되지 않는다.

❖ 부모 클래스의 메소드에 덧붙여서 소스 코드를 확장하고 싶은 경우도 생긴다.

- 이때 부모 클래스의 메소드 내용을 Copy & Paste 하는 것은 또 다른 소스 코드를 중복을 야기하게 된다.
- 이런 경우를 위해 파이썬은 `super()`라는 함수를 제공

✓ `super()` : 부모 클래스의 인스턴스가 호출이 되어 부모 클래스의 속성이나 메소드를 호출