Ask Django

오버로딩과 오버라이딩

오버로딩과 오버라이딩 #wikipedia

- 오버로딩 (Overloading)
  - 이름은 같지만, 인자와 리턴타입이 다른 멤버 함수를 여럿 정의하는 것.
  - 파이썬에서는 미지원
- 오버라이딩 (Overriding)
  - 클래스 상속에서 사용되는 개념
  - 상위 클래스가 가지고 있는 메소드를 하위 클래스가 재정의

Overloading 예시

Sample.java

같은 이름 calculate의 인스턴스 함수를 여럿 정의할 수 있습니다.

```
class Sample {
    public static int calculate(int x, int y) {
        return x + y;
    }

    public static int calculate(int x, int y, int z) {
        return (x + y) * z;
    }

    public static void main(String[] args) {
        int result1 = calculate(1, 2);
        int result2 = calculate(1, 2, 3);
        System.out.println(String.format("%d, %d", result1, result2));
    }
}
```

실행

```
쉘> javac Sample.java && java -classpath . Sample
3, 9
```

Sample.py

대신, 디폴트인자를 통한 처리가 가능합니다.

```
class Sample:
    def calculate(self, x, y, z=1):
        return (x + y) * z

>>> sample = Sample()
>>> sample.calculate(1, 2), sample.calculate(1, 2, 3)
(3, 9)
```

오버라이딩 (Overriding)

클래스 주요 오버라이딩 멤버함수

- __init__(self[, ...]) : 생성자 함수 #doc
- __repr__(self) : 시스템이 해당객체를 인식할수있는 Official 문자열 #doc
  - 대개 디버깅을 위해 사용
  - 출력 문자열을 통해, 바로 인스턴스를 생성할 수 있도록, 인스턴스 생성
- __str__(self) : Informal 문자열. str(obj) 시에 호출
- __getitem__(self, key) : self[key] 를 구현 #doc
- __setitem__(self, key, value) : self[key] =value #doc

클래스 주요 오버라이딩 멤버함수 - 연산자 재정의 #ref

- Binary Arthmetic Operations
  - +, -, *, @, /, //, %, divmod, pow, **, <<, >>, &, ^, |
  - ex) **x + y** 는 **x.__add__(y)** 함수를 호출
- Augmented Arithmetic Assignments
  - +=, -=, *=, @=, /=, //=, %=, **=, <<=, >>=, &=, ^=, |=
  - ex) **x += y** 는 **x.__iadd__(y)** 함수를 호출
- Unary Arthmetic Operations : - , + , abs, ~
  - ex) **-obj**는 **obj.__neg__()** 함수를 호출
- built-in functions : complex, int, float, round
  - ex) **complex(obj)**는 **obj.__complex__()** 함수를 호출
- Rich Comparison : <, <=, ==, !=, >, >=
  - ex) **x < y**는 **x.__lt__(y)** 함수를 호출

예시 : __add__, __iadd__ 구현하기

```
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age

    def __add__(self, value):
        return Person(self.name, self.age + value)

    def __iadd__(self, value):
        self.age += value
        return self

    def __repr__(self):
        return "Person('{}', {})".format(self.name, self.age)

>>> tom = Person('Tom', 10)

>>> tom + 10
Person('Tom', 20)

>>> tom += 20

>>> tom
Person('Tom', 30)
```

클래스 주요 오버라이딩 멤버함수 - with절 지원

- __enter__(self) #ref
- __exit__(self, exc*type*, *exc*value, traceback) #ref
  - exc_type : 예외 (Exception) 클래스 타입
  - exc_value : 예외 인스턴스
  - traceback : Traceback 인스턴스
  - 예외가 발생하지 않았다면, 인자 3개 값은 모두 None으로서 호출

예시 : 클래스를 통한 with절 지원

```
class File:
    def __init__(self, path, mode):
        self.path = path
        self.mode = mode

    def __enter__(self):
        self.f = open(self.path, self.mode, encoding='utf-8')
        return self.f

    def __exit__(self, exc_type, exc_value, traceback):
        # 예외 발생여부에 상관없이 파일을 닫습니다.
        self.f.close()

with File('filepath.txt', 'wt') as f:
    f.write('hello world')
```

물론 파이썬 기본에서도 지원하고 있습니다. ㅎㅎ :D

```
with open('filepath.txt', 'wt') as f:
    f.write('hello world')
```