Python 객체

키워드 : [Python 객체 & 클래스 (https://docs.python.org/ko/3.7/tutorial/classes.html)](https://docs.python.org/ko/3.7/tutorial/classes.html)

우리는 자바스크립트 언어에 대해서 공부할 때, '객체'라는 것에 대해서 이야기했습니다. 사실, 자바스크립트뿐만 아니라, 대부분의 프로그래밍 언어가 '객체'라는 개념을 도입하여 사용할 수 있습니다. (Java, Javascript, C++, Python, etc.) 다만, 객체를 선언하고, 사용하는 방법에 약간씩 차이는 있습니다.

하지만, 대부분의 내용이 대동소이하므로, 하나의 프로그래밍 언어에서 개념을 잘 숙지해두시면 다른 프로그래밍 언어를 이용해 개발을 진행해도, 금세 익숙해지실 수 있을 거라 생각합니다. 특히나, 대부분의 프로그래밍 언어에서 사용되는 객체와 관련된 개념과 기능을 파이썬 언어에서도 똑같이 제공하는 것으로 알려져 있습니다. 때문에 파이썬 언어의 객체를 잘 이해하신다면, 다른 프로그래밍 언어에서 객체를 선언하고, 사용하기 수월하겠죠? :)

이번 장에서는 파이썬 언어에서 객체를 선언하고, 사용하는 방법에 대해서 알아보도록 하겠습니다.

클래스

우리는 강의를 통해 파이썬에서는 클래스라는 것을 선언하고, 사용했습니다. 자바스크립트에서 살펴보았던 것처럼 리터럴 (`var obj = {Properties...}`) 형태로 작성될 순 없습니다.

그렇다면, 클래스라는 것은 무엇이고 왜 사용되는 것일까요? 우선, 클래스는 그 자체로 객체입니다. 객체라는 것은 의미있는 것들을 묶어, 하나의 개념으로 '추상화' 시킨 것이라고 할 수 있습니다. 클래스는, 이러한 추상화된 개념을 담는 '틀' 혹은 '형태' 등으로 바꿔 표현할 수 있습니다.

조금 더 프로그래밍 적으로 접근하면, 데이터(속성)와 기능(메서드)들을 하나로 묶기 위해 사용되는 전용 키워드쯤으로 생각할 수 있습니다. 예를 들어, 고양이를 프로그래밍 코드로 표현하고 싶다고 가정해보도록 하겠습니다.

```
# 기본 형식
class 클래스명():
    # ...

# 1번 코드
class Cat():
    # 2번 코드
    species = '고양이과'
    # 3번 코드
    def __init__(self, name):
        self.play = []
        self.name = name
    # 4번 코드
    def add_play(self, game):
        self.play.append(game)

# 5번 코드
cat1 = Cat('냥1')
cat2 = Cat('냥2')
# 6번 코드
print(cat1.name, cat2.name) # 냥1 냥2
cat1.add_play('사냥놀이')
print(cat1.play) # ['사냥놀이']
print(cat1.species, cat2.species) # 고양이과 고양이과
# 7번 코드
Cat.species = '개과'
# 8번 코드
print(cat1.species) # '개과'
print(cat2.species) # '개과'
```

위 코드를 통해 클래스에 대한 내용을 하나하나 살펴보도록 하겠습니다. 위 코드 내용 중 아직은 이해하기 어려운 def 키워드의 경우, 파이썬에서 함수를 선언하기 위해 사용되는 키워드로 생각해주시면 됩니다.

1. 1번 코드는 `class Cat():`코드를 통해서 Cat이라는 이름을 갖는 객체를 선언하고 있습니다. 이런 식으로, "클래스명"은 객체의 이름을 붙여주는 작업입니다. 파이썬 코드 어디서든, 이 이름을 호출해서 사용할 수 있습니다.

2. 2번 코드는 species라는 변수를 선언하고, 사용하고 있습니다. 일반적으로, 어떤 메서드(def 문) 안에 작성된 코드가 아닌, 클래스 이름 바로 밑에 작성된 변수는 "클래스 속성(Attribute)"이 되며, 모든 객체가 공유하는 값이 됩니다. 이 내용은, 7번 코드에서 조금 더 이야기하도록 합니다.

3. 3번 코드의 `__init__` 키워드는 상당히 중요한데요, 우리가 객체를 선언하고 사용할 때, 해당 객체에 속해있는 여러 변수(Attribute)를 초기화하고 사용하게 됩니다. 이 초기화를 담당하는 키워드가 `def __init__(self)` 키워드입니다. 초기화를 하는 방법은, 인자를 받아오는 방법(`self.name = name`)과 생성자 안에서 바로 빈 값을 만들어 초기화(`self.play = []`)하는 방식이 있습니다. (인자의 첫 번째는 무조건 `'self'` 키워드가 와야 합니다.)

4. 4번 코드는 add_play라는 이름의 메서드를 선언하고, 생성자를 통해 초기화한 `self.game` 리스트에 인자로 넘어온 값을 할당하는 코드 입니다. (여기서 `'append'`라는 키워드는 리스트 가장 마지막 인덱스에 값을 하나하나 추가하는 코드입니다.)

5. 5번 코드는 cat1이라는 인스턴스와 cat2라는 두 가지 인스턴스를 생성하고 있습니다. 여기서 이야기하는 인스턴스라는 것은, 인스턴스명 = 클래스명() 형태로 작성된 클래스의 속성과 메서드를 사용할 수 있는 하나의 변수를 나타냅니다.

6. 6번 코드에서는 인스턴스의 속성(Attribute)의 이름을 닷(.) 연산자를 이용해 접근하고 있습니다. 각각의 인스턴스는 메모리 영역에서 서로 다른 위치에 자리잡고 있으며, 서로 다른 속성을 같습니다.

7. 7번 코드는 인스턴스 이름이 아닌, 클래스이름(Cat)으로 species라는 속성에 접근하고 있습니다. 이러한 경우를 보통 어트리뷰트 참조(Attribute reference)라고 부릅니다.

8. 7번 코드를 통해 클래스 속성의 값을 변경하면, 8번 코드와 같이 Cat 클래스의 인스턴스는 모두 해당 속성의 값을 공유하게 됩니다.

위에 내용 중에 사용된 생성자(def __init__(self))의 경우엔, 클래스 선언시에 꼭 필요한 키워드는 아닙니다. 다만, 어떤 객체를 선언하고 사용할 때, 객체의 초기화 작업은 이 생성자를 호출해서 진행하는 것이 코드를 관리하기 수월합니다. 또한, 클래스 안에 모든 메서드가 인스턴스 메서드일 필요는 없습니다.

cls 키워드를 첫 번째 인자로 갖는 클래스 메서드로 작성할 수도 있으며, 아무런 키워드도 붙지 않는 일반 함수처럼 작성하고, 사용할 수도 있습니다.

참고로, 클래스 이름의 시작은 영문 대문자로 시작합니다. 만약, 여러 단어를 하나로 조합해야 할 경우에는 단어의 첫 단어를 대문자로 작성합니다. e.g., DataFilter, ImageProcessing, etc.

그렇다면, 클래스를 언제 사용할까요? 클래스를 언제, 어디에, 어떻게 선언하고, 사용할 수 있을지와 관련된 내용은 사실, 간단히 이야기하기 어렵습니다. 클래스를 어떻게 선언하고, 객체를 어떻게 관리할지 등을 정형화된 패턴으로 정리한 '디자인 패턴 (https://github.com/faif/python-patterns)'이라는 것도 존재할 정도입니다. (단기간에 익히기엔 난이도가 상당히 있습니다.)

더욱이 클래스와 객체라는 것은 상속, 다형성과 같은 많은 심화 내용이 존재합니다. 또한, 메모리 영역과 굉장히 밀접한 관계를 같기 때문에, 성능 이슈와 같은 꽤나 머리 아픈? 내용과도 연결됩니다. 이러한 내용들은, 한 번에 이해하기엔 어렵고, 공부하기에 딱딱한 내용이 아닐 수 없습니다. 기본적인 개념들을 탄탄히 쌓고, 다양한 관련 서적과 참고 자료 등을 찾아보며, 조금씩 이해하려 노력해보세요! :)