파이썬 객체

2019년 6월 7일 금요일 오후 2:21

객체지향

- 프로그램은 서로 협력하는 여러 개의 객체로 구성
- "전체 프로그램"이 아닌 각각의 객체가 마치 프로그램안의 "섬"같이 서로 협력하여 작동
- 프로그램은 함께 실행되는 여러 개의 객체로 구성 - 객체는 서로의 기능들을 활용

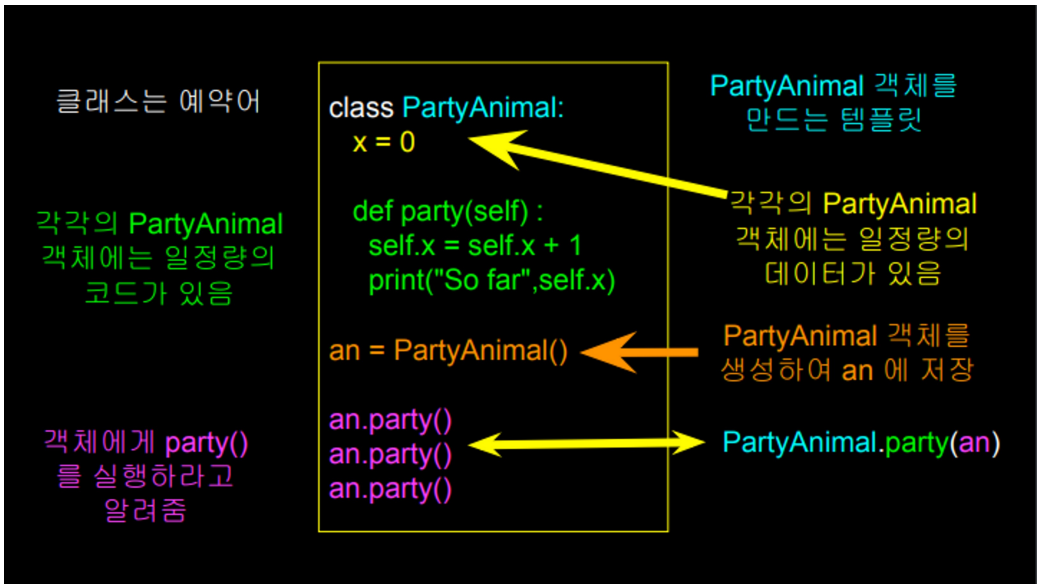
객체

- 객체는 하나의 자족적인 코드와 데이터
- 객체 지향 접근의 요점은 제를 이해가능한 작은 문제로 분할하여 접근(분할 정복 divide-and-conquer)
- 객체는 우리가 필요없는 세부사항들을 무시할 수 있도록 경계를 제공
- 우리는 객체를 계속 사용해 왔음 : 문자열 객체, 정수형 객체, 딕셔너리 객체, 리스트 객체
- 객체는 세부사항을 감춤 - "프로그램의 나머지 부분"의 세부사항을 무시할 수 있게 해줌

<정의>

클래스 Class	하나의 형식, 템플릿
메소드 Method or Message	클래스 내에 정의된 기능
필드/속성 Field or attribute	클래스 내의 데이터
객체/인스턴스 Object or Instance	클래스의 한 인스턴스

❖ **클래스와 메소드의 정의, 사용법**



결과

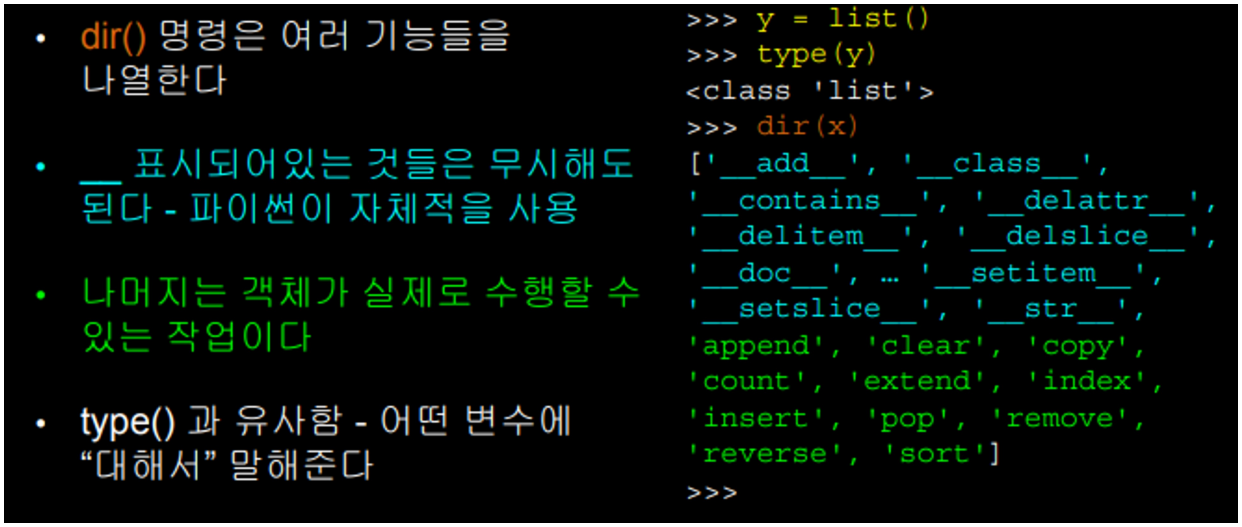
```
$ python party1.py
So far 1
So far 2
So far 3
```

- PartyAnimal 이라는 이름의 class 에 변수 x와 party() 메소드를 정의함.

- party()메소드에 매개변수로는 자기자신의 객체 self가 전달됨

❖ **dir() 과 type()**

dir 함수와 type 함수를 사용하면 객체를 검사할 수 있다.



- 각 객체에 dir 함수를 실행시키면 해당 객체에서 사용 가능한 메소드와 기본으로 포함되는 변수들을 알 수 있습니다.
  x는 리스트 객체이기 때문에 append, sort 등의 메소드를 실행시킬 수 있다는 것을 알 수 있습니다.
- type 함수에 y객체를 넣으면 y가 어떤 클래스에 속한 객체인지를 알 수 있습니다.
  다음 코드에서는 y가 리스트의 객체라는 것을 우리에게 알려줍니다.

✓ **직접 만든 클래스의 객체 검사하기**

```
class PartyAnimal:
    x = 0

    def party(self) :
        self.x = self.x + 1
        print("So far",self.x)

an = PartyAnimal()

print("Type", type(an))
print("Dir ", dir(an))

# Type <class '__main__.PartyAnimal'>
# Dir  ['__class__', '__delattr__', '__dict__', '__dir__', '__doc__', '__eq__', '__format__', '__ge__', '__getattribute__', '__gt__',
# '__hash__', '__init__', '__init_subclass__', '__le__', '__lt__', '__module__', '__ne__', '__new__', '__reduce__', '__reduce_ex__',
# '__repr__', '__setattr__', '__sizeof__', '__str__', '__subclasshook__', '__weakref__',
# 'party', 'x']
```

- type 함수를 통해 an이라는 객체는 PartyAnimal타입의 클래스 객체라는 것을 알 수 있다.
  __main__는 an객체의 범위이다.
- dir 함수를 통해 an 객체에는 party라는 메소드와 x라는 변수가 포함되어있다는 사실을 알 수 있다.
  an.x an.party 등으로 사용될 수 있다.

✓ **dir()을 문자열에 사용**

```
>>> x = 'Hello there'
>>> dir(x)
['__add__', '__class__', '__contains__', '__delattr__',
'__doc__', '__eq__', '__ge__', '__getattribute__',
'__getitem__', '__getnewargs__', '__getslice__', '__gt__',
'__hash__', '__init__', '__le__', '__len__', '__lt__',
'__repr__', '__rmod__', '__rmul__', '__setattr__', '__str__',
'capitalize', 'center', 'count', 'decode', 'encode', 'endswith',
'expandtabs', 'find', 'index', 'isalnum', 'isalpha', 'isdigit',
'islower', 'isspace', 'istitle', 'isupper', 'join', 'ljust',
'lower', 'lstrip', 'partition', 'replace', 'rfind', 'rindex',
'rjust', 'rpartition', 'rsplit', 'rstrip', 'split',
'splitlines', 'startswith', 'strip', 'swapcase', 'title',
'translate', 'upper', 'zfill']
```

파이썬 객체의 생명주기

2019년 6월 7일 금요일 오후 3:57

객체 생명주기

- 객체는 생성되고, 사용되어지고, 없어짐
- 객체를 불러올 때 특별한 코드(메소드)가 존재
  - 생성되어 질 때 (생성자)
  - 소멸되어 질 때 (소멸자)
- 성자는 자주 사용
- 소멸자는 거의 사용되지 않음

❖ 생성자

- 생성자의 주된 목적은 인스턴스 변수가 객체가 생성될 때 적절한 초기값을 가지게 하는 것.
- 객체지향 프로그래밍에서, 클래스의 생성자는 객체가 생성되어질 때 불러오는 문장.

❖ 생성자(Constructor)와 소멸자(Destructor)

파이썬의 생성자는 __init__, 소멸자는 __del__로 정의합니다.

이 코드에서는 객체를 생성할 때 생성자가 실행이 되고, 객체가 사라질 때 소멸자가 실행되는 것을 볼 수 있습니다.

```python
class PartyAnimal:
    x = 0

    def __init__(self):
        print('I am constructed')

    def party(self) :
        self.x = self.x + 1
        print('So far',self.x)

    def __del__(self):
        print('I am destructed', self.x)

an = PartyAnimal()
an.party()
an.party()
an = 42
print('an contains',an)

# I am constructed
# So far 1
# So far 2
# I am destructed 2
# an contains 42
```

-> 생성자

-> 소멸자

객체가 생성될 때 생성자가 실행됨

생성된 객체를 담고있던 변수에 다른 값을 대입하면, 기존의 객체를 덮어쓰게 되고, 기존의 객체는 소멸된다. 따라서 소멸자가 실행됨

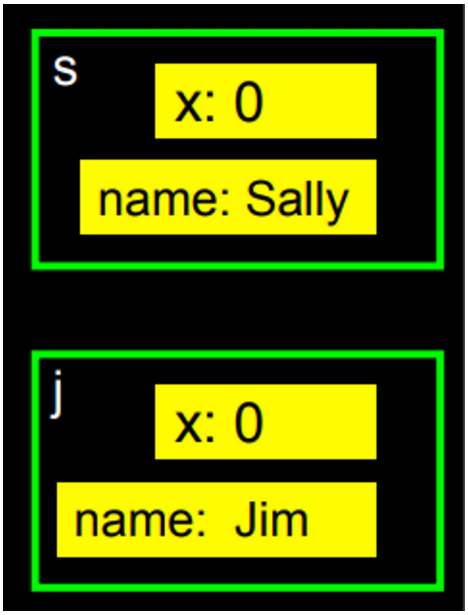
여러 인스턴스

- 다수의 객체를 만들 수 있음 - 클래스가 그 형식의 틀을 제공
- 별개의 객체들을 각자의 변수에 저장할 수 있음
- 이것을 동일 클래스의 다중 인스턴스 라고 함
- 각각의 인스턴스는 자신만의 인스턴스 변수를 가지고 있음

✓ **생성자에 매개변수 추가해보기**

```python
class PartyAnimal:
    x = 0
    name = ""
    def __init__(self, z):
        self.name = z
        print(self.name,"constructed")

    def party(self) :
        self.x = self.x + 1
        print(self.name,"party count",self.x)

s = PartyAnimal("Sally")
j = PartyAnimal("Jim")

s.party()
j.party()
s.party()

# Sally constructed
# Jim constructed
# Sally party count 1
# Jim party count 1
# Sally party count 2
```

이 코드의 결과를 보면 객체를 두 번 생성할 경우 name이라는 속성과 x라는 속성은 각각 별개의 값을 저장한다는 걸 알 수 있습니다.



파이썬 상속

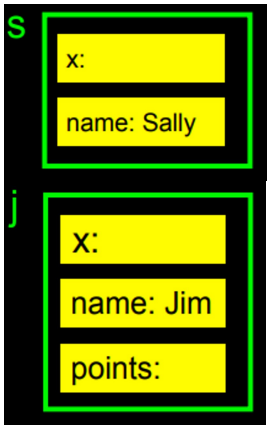
2019년 6월 7일 금요일 오후 4:43

상속

- 새로운 클래스를 만들 때 - 기존에 있던 클래스를 재사용하여 그 모든 기능을 상속받고 조금 추가하여 새로운 클래스를 만들 수 있음
- 저장하고 재사용하는 또 다른 형태
- 한 번 작성하고 - 여러 번 재사용
- 새로운 클래스는 (child) 이전 클래스(parent) 의 모든 기능을 가지고 있음
  - 그리고 더 추가할 수 있음

✓ **PartyAnimal 클래스와 PartyAnimal 클래스를 상속한 FootballFan 클래스를 정의하고, 각각의 객체를 생성하여 실행하는 코드입니다.**

```
1  class PartyAnimal:
2      x = 0
3      name = ""
4      def __init__(self, nam):
5          self.name = nam
6          print(self.name,"constructed")
7
8      def party(self) :
9          self.x = self.x + 1
10         print(self.name,"party count",self.x)
11
12 class FootballFan(PartyAnimal):
13     points = 0
14     def touchdown(self):
15         self.points = self.points + 7
16         self.party()
17         print(self.name,"points",self.points)
18
19 s = PartyAnimal("Sally")
20 s.party()
21
22 j = FootballFan("Jim")
23 j.party()
24 j.touchdown()
25
26 # Sally constructed
27 # Sally party count 1
28 # Jim constructed
29 # Jim party count 1
30 # Jim party count 2
31 # Jim points 7
```



-> 클래스를 정의할 때 괄호안에 PartyAnimal을 넣음으로써 FootballFan 클래스는 PartyAnimal 클래스를 상속받는다.

-> j객체는 s객체보다 더 많은 것을 가지고 있다.

-> FootballFan 객체를 생성할 때 PartyAnimal 객체가 먼저 성되어 PartyAnimal 의 init 생성자가 실행된다.

-> 실행결과를 보면 j는 FootballFan의 객체이지만 PartyAnimal 클래스의 생성자와 party 메소드 등을 모두 상속한 것을 알 수 있다.

➢ **정의**

- 클래스 Class - 형식 또는 템플릿
- 속성 Attribute – 클래스 내의 변수
- 메소드 Method - 클래스 내의 함수
- 객체 Object - 클래스의 특정 인스턴스
- 생성자 Constructor – 객체가 생성될 때 실행되는 코드
- 상속 Inheritance - 기존의 클래스를 확장하여 새로운 클래스를 만드는 것