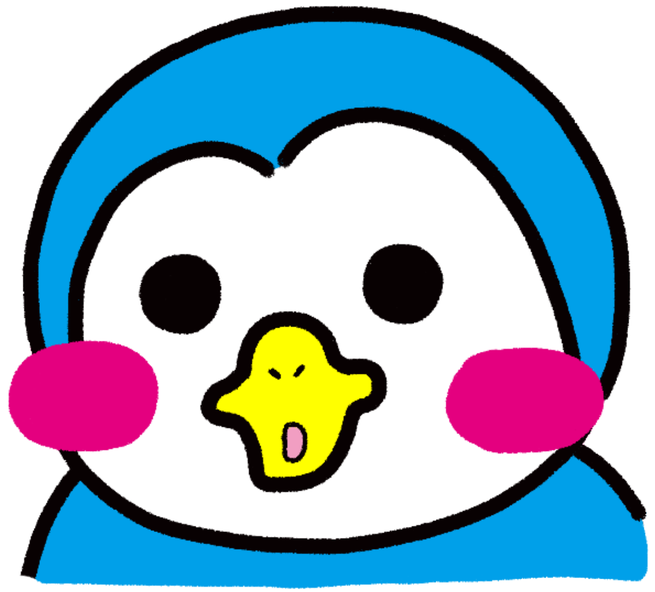
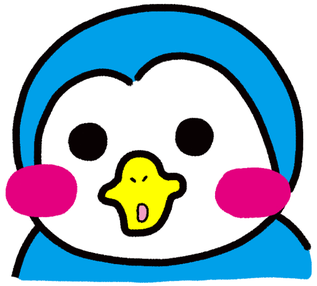
입문자를 위한 파이썬 기초



self의 역할과 생성자의 소개

클래스는 하나, 객체는 여러 개



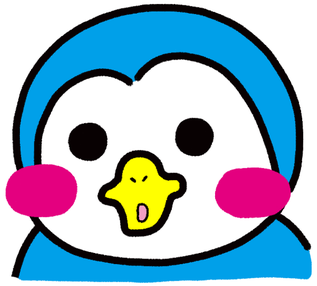
하나의 클래스(자료형)가 정의되어 있으면, 해당 클래스를 기반으로 여러 개의 객체를 생성할 수 있다. 이때 중요한 사실은, 같은 클래스를 기반으로 생성했다 해도 각각의 객체는 서로 다른 메모리 영역을 할당받은 독립적인 데이터라는 사실이다.

```
class Cat :
  def meow(self) :
    print("야옹~~")


myCat1 = Cat()
myCat2 = Cat()
```

myCat1 그리고 myCat2 객체는
자료형만 같을 뿐 서로 다른 객체이다!

self 너는 누구냐



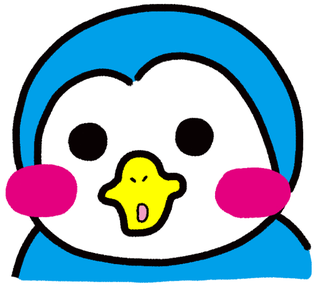
클래스 정의 구문에서 self 매개변수는 '현재 사용 중인 객체'를 의미한다!

```
class Cat :
  def meow(self) :
    print("야옹~~")

myCat1 = Cat()
myCat2 = Cat()

myCat1.meow() # self는 myCat1을 뜻한다
myCat2.meow() # self는 myCat2를 뜻한다
```

self 너는 누구냐



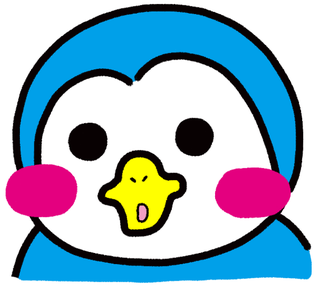
클래스 정의 구문에서 self 매개변수는 '현재 사용 중인 객체'를 의미한다!

```
class Cat :
    def meow(self) :
        print("야옹~~")

myCat1 = Cat()
myCat2 = Cat()

myCat1.meow() # self는 myCat1을 뜻한다
myCat2.meow() # self는 myCat2를 뜻한다
```

self 너는 누구냐



클래스 정의 구문에서 self 매개변수는 '현재 사용 중인 객체'를 의미한다!

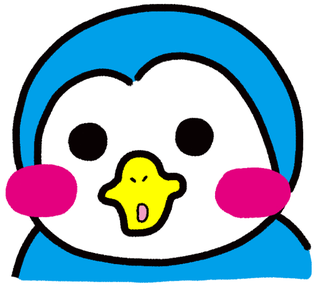
```
class Cat :
    def meow(self) :
        print("야옹~~")

myCat1 = Cat()
myCat2 = Cat()

myCat1.meow() # self는 myCat1을 뜻한다
myCat2.meow() # self는 myCat2를 뜻한다
```

어차피 객체 그 자신을 의미하므로
따로 인수를 전달해 줄 필요가 없다!

자신만의 것을 가질 수 있다



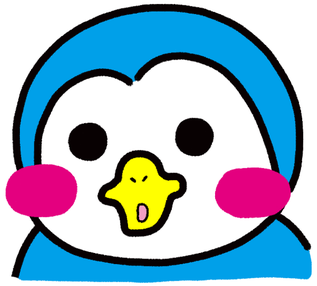
self는 '현재 사용 중인 객체'를 의미한다. self를 이용하면 각 객체가 자신만의 상탯값을 따로 가질 수 있다.

```
class Cat :
  def setName(self, arg1) :
    self.name = arg1

myCat1 = Cat()
myCat2 = Cat()

# self는 정해져 있기 때문에, arg1만 전달하면 된다!
myCat1.setName("귀요미") # myCat1.name = "귀요미"
myCat2.setName("야옹이") # myCat2.name = "야옹이"
```

자신만의 것을 가질 수 있다



self는 '현재 사용 중인 객체'를 의미한다. self를 이용하면 각 객체가 자신만의 상탯값을 따로 가질 수 있다.

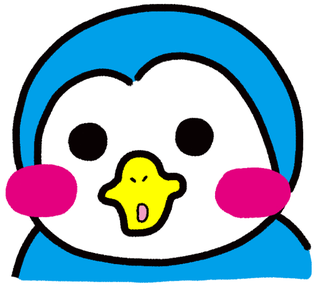
```
class Cat :
  def setName(self, arg1) :
    self.name = arg1

myCat1 = Cat()
myCat2 = Cat()

# self는 정해져 있기 때문에, arg1만 전달하면 된다!
myCat1.setName("귀요미") # myCat1.name = "귀요미"
myCat2.setName("야옹이") # myCat2.name = "야옹이"
```

전달받은 인수를
현재 사용 중인 객체의 상탯값
name에 초기화하겠다!

자신만의 것을 가질 수 있다



self는 '현재 사용 중인 객체'를 의미한다. self를 이용하면 각 객체가 자신만의 상탯값을 따로 가질 수 있다.

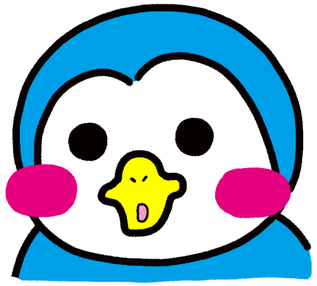
```
class Cat :
  def setName(self, arg1) :
    self.name = arg1

myCat1 = Cat()
myCat2 = Cat()

# self는 정해져 있기 때문에, arg1만 전달하면 된다!
myCat1.setName("귀요미") # myCat1.name = "귀요미"
myCat2.setName("야옹이") # myCat2.name = "야옹이"
```

같은 클래스 기반으로 만든 두 객체는 모두 name이라는 상태를 가지지만, 두 객체의 name 값은 서로 다르다!

자신만의 것을 가진 채 생성된다

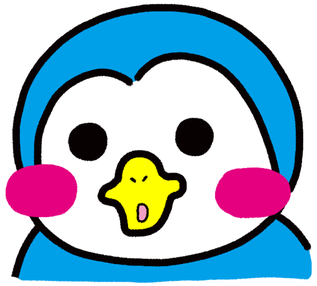


클래스 내부에 생성자(constructor)를 정의하면 객체는 생성되는 시점부터 자신만의 상탯값을 가질 수 있다. 생성자의 이름은 정해져 있는 것을 사용한다.

```
class Cat :
  def __init__(self, arg1) :
    self.name = arg1

myCat1 = Cat("귀요미") # myCat1.name = "귀요미"
myCat2 = Cat("야옹이") # myCat2.name = "야옹이"
```

자신만의 것을 가진 채 생성된다

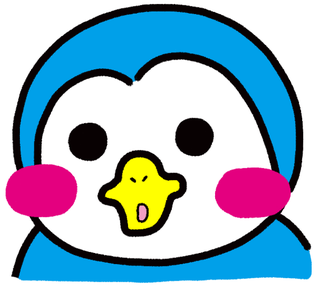


클래스 내부에 생성자(constructor)를 정의하면 객체는 생성되는 시점부터 자신만의 상탯값을 가질 수 있다. 생성자의 이름은 정해져 있는 것을 사용한다.

```
class Cat :
  def __init__(self, arg1) :
    self.name = arg1

myCat1 = Cat("귀요미") # myCat1.name = "귀요미"
myCat2 = Cat("야옹이") # myCat2.name = "야옹이"
```

자신만의 것을 가진 채 생성된다



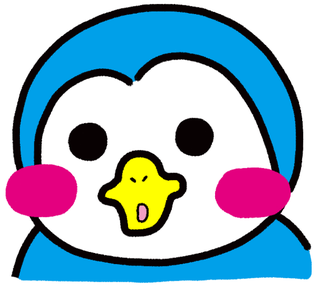
클래스 내부에 생성자(constructor)를 정의하면 객체는 생성되는 시점부터 자신만의 상탯값을 가질 수 있다. 생성자의 이름은 정해져 있는 것을 사용한다.

```
class Cat :
  def __init__(self, arg1) :
    self.name = arg1

myCat1 = Cat("귀요미") # myCat1.name = "귀요미"
myCat2 = Cat("야옹이") # myCat2.name = "야옹이"
```

=> 객체 생성 시 클래스 이름 옆에 붙여주는 괄호는 생성자를 위한 것이었다!

내용 정리



- 정의된 클래스를 기반으로 여러 개의 객체를 생성할 수 있다.
- 같은 클래스 기반으로 생성했다 해도 각각의 객체는 서로 다른 메모리 영역을 할당받은 독립적인 데이터이다.
- 클래스 정의 구문에서 self 매개변수는 '현재 사용 중인 객체'를 의미한다.
- self를 이용하면 각 객체가 자신만의 상탯값을 따로 가질 수 있다.
- 생성자를 정의하면 객체는 생성되는 시점부터 자신만의 상탯값을 가질 수 있다. 생성자는 객체 생성 시점에 동작하는 메소드이다.