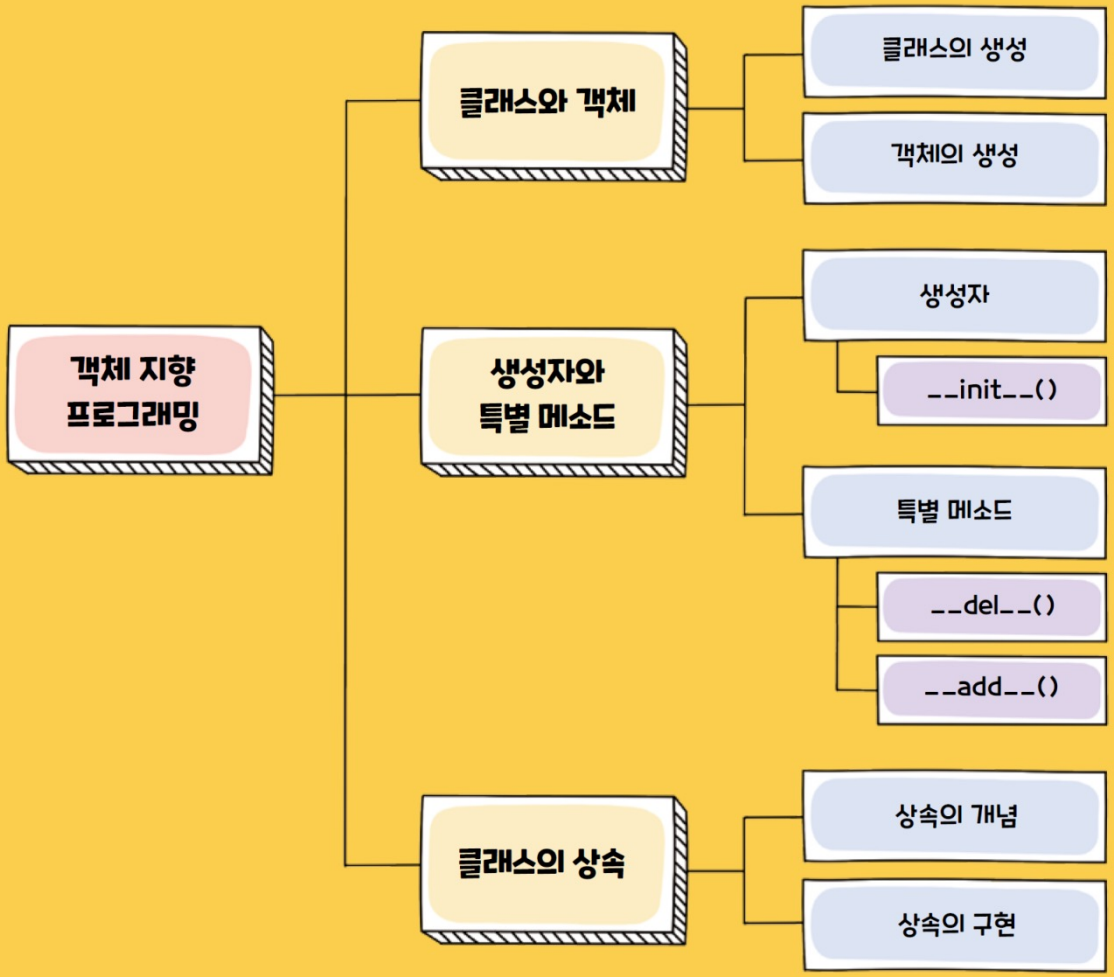
Chapter 10
객체지향 프로그래밍

임경태

Preview


객체 지향
프로그래밍
클래스와 객체
클래스의 생성
객체의 생성
생성자와
특별 메소드
생성자
__init__()
특별 메소드
__del__()
__add__()
클래스의 상속
상속의 개념
상속의 구현

학습목표

- 객체의 개념을 파악합니다.
- 파이썬에서 제공하는 객체를 이해합니다.
- 새로운 객체를 생성하는 방법을 익힙니다.
- 생성자와 클래스 상속의 개념을 학습합니다.

Section 01
객체 지향 프로그래밍 이해하기

객체의 개념

객체

- 수많은 사물을 프로그래밍 관점에서 객체(Object)라고 부름
  - 자동차, 건물, 고양이, 물고기 등
- 객체의 정의

> 객체란 어떤 속성과 행동을 가지고 있는 데이터를 말한다.

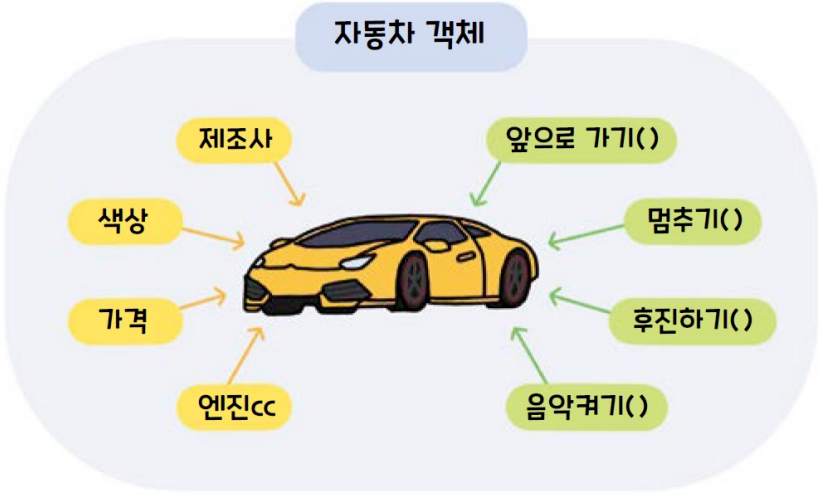


그림 10-1 자동차의 속성과 행동

이미 사용해 봤던 객체들

거북이 객체

```
>>> import turtle
>>> turtle.shape('circle')
>>> turtle.shape()
'circle'
>>> turtle.goto(100,100)
>>> turtle.xcor()
100
>>> turtle.ycor()
100
```

문자열 객체

```
>>> myStr = "My first Life"
>>> myStr.upper()
‘My first Life'
>>> myStr.count('i')
2
>>> myStr.isalpha()
True
```

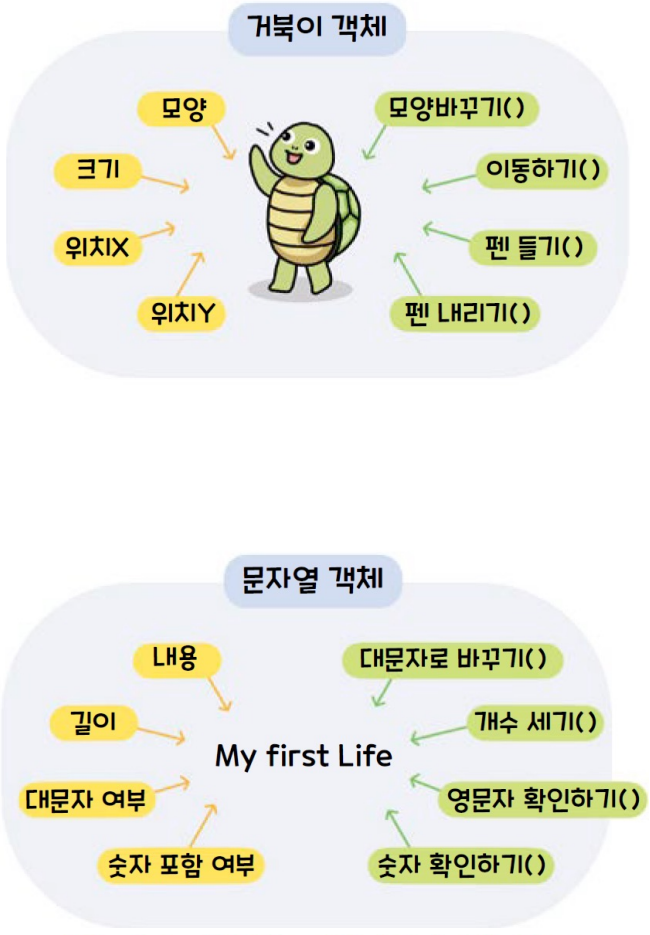


그림 10-2 거북이와 문자열 객체의 속성과 행동

이미 사용해 봤던 객체들

확인문제

다음 빈칸에 들어갈 단어를 채우시오.

지금까지 사용한 거북이 및 문자열 등은 객체이다. 객체는 _____(와)과 _____(으)로 이루어져 있다.

정답

Click!

Section 02
클래스와 객체

클래스와 객체의 관계

클랫스

- 파이썬에서 제공하는 객체 외에 필요한 객체를 만들어서 사용할 수 있는데, 객체를 만들기 위해서는 클래스가 필요함
- 객체를 만들기 위한 설계도 또는 찍어내는 틀을 의미함

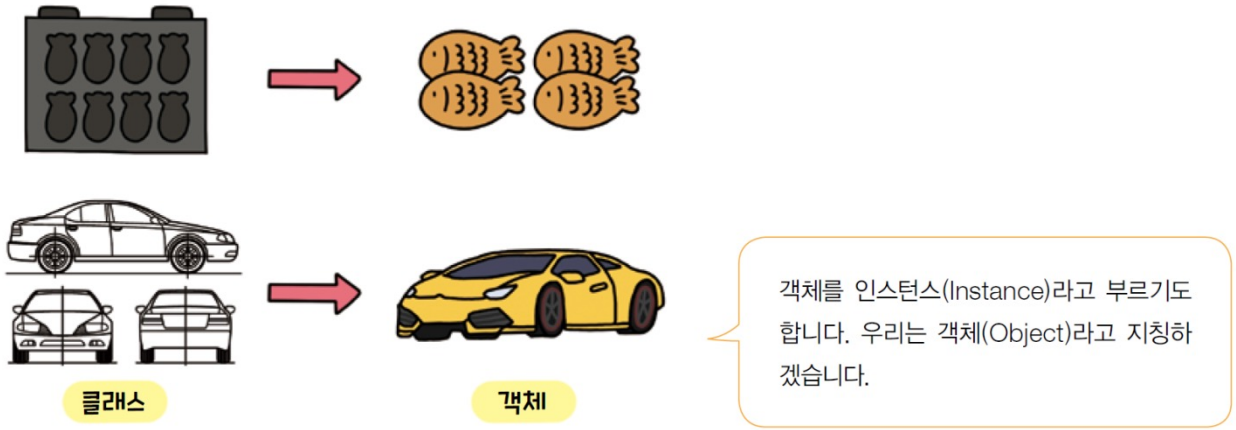


그림 10-3 클래스와 객체의 개념

클래스의 생성

- 파이썬이 제공하지 않는 토끼 클래스 만들기
  - 토끼 객체를 만들려면 먼저 토끼 클래스가 필요함
  - 하지만 토끼 클래스는 파이썬에서 제공하지 않음
    - 따라서 토끼 객체를 만들기 위해서는 우선 토끼 클래스를 만들어야 함

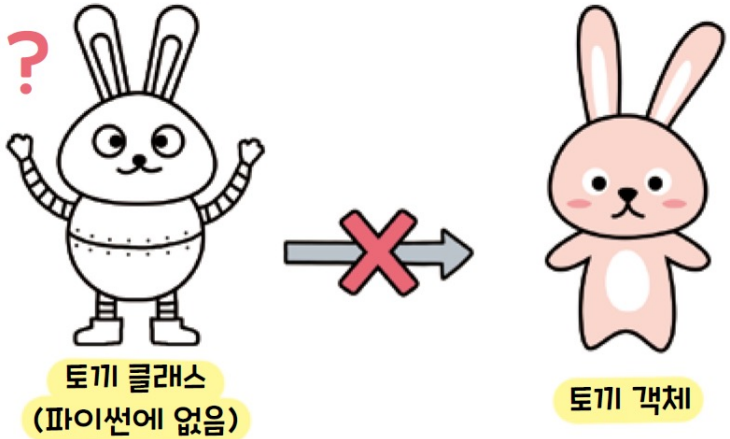


그림 10-4 토끼 객체 생성 못함

클래스의 생성

클래스를 만들기 위한 형식

```
class 클래스이름 :
    # 클래스 생성 코드 구현
```

- 토끼 클래스에 필요한 속성과 행동
  - 속성 : 토끼의 모양, X, Y 위치 → 변수로 생성
  - 기능 : 이동하기( ) → 함수(메소드)로 구현

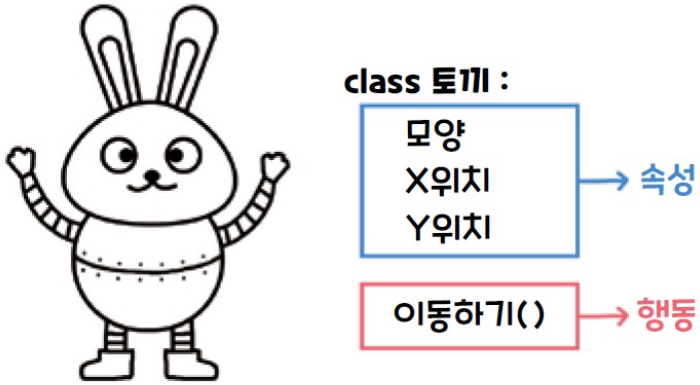


그림 10-5 토끼를 클래스로 구현

클래스의 생성

- 토끼 클래스를 좀 더 구체적으로 구현하기

```
class Rabbit :
    # 토끼의 속성(변수)
    모양 = ""
    X위치 = 0
    Y위치 = 0

    # 토끼의 행동(메소드)
    def goto(이동할 좌표) :
        # 토끼를 이동할 좌표로 이동시키는 코드
```

- 토끼 클래스를 파이썬 코드로 구현하기

[코드 10-1]

```
class Rabbit :
    shape = "" # 토끼 모양
    xPos = 0   # X위치
    yPos = 0   # Y위치

    def goto(self, x, y) :
      self.xPos = x
      self.yPos = y
```

객체의 생성

토끼 객체 생성

- 토끼 설계도(클래스)로, 실제 토끼(객체)를 제작하는 작업이 필요함

그림 10-6 클래스와 객체 개념

```
rabbit1 = Rabbit()
rabbit2 = Rabbit()
rabbit3 = Rabbit()
```

객체의 생성

속성에 값 대입하기

- 각각의 토끼는 서로 별도의 토끼임
  - 즉 컴퓨터 안에서 서로 다른 메모리 공간을 차지함
- 각 객체에는 별도의 속성이 존재하고, 각각에 별도의 값을 대입할 수 있음
- 각각의 토끼는 **객체이름.속성이름**으로 접근할 수 있음

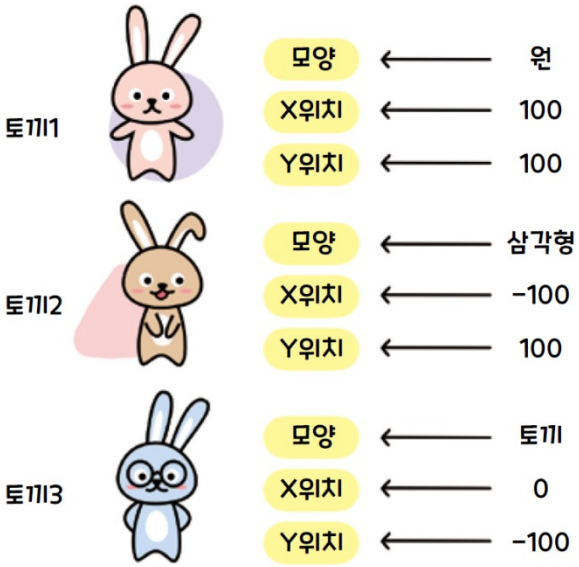


그림 10-7 객체의 속성에 값을 대입하는 개념

```
rabbit1.shape = "원"
rabbit2.shape = "삼각형"
rabbit3.shape = "토끼"
```

객체의 생성

- 메소드의 호출
  - 메소드도 각 객체마다 별도로 존재함
  - 메소드의 호출은 **객체이름.메소드이름()** 형식을 사용함

```
rabbit1.goto(100, 100)
rabbit2.goto(-100, 100)
rabbit3.goto(0, -100)
```

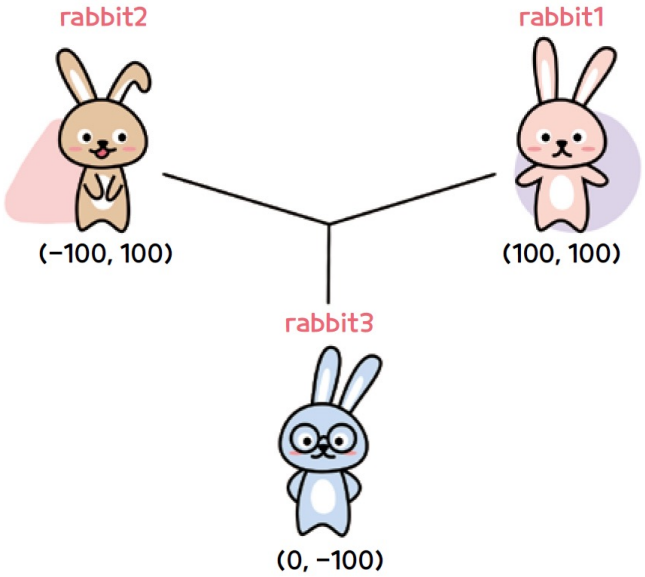


그림 10-8 토끼의 속성 변경 및 행동(메소드) 사용 결과

객체의 생성

확인문제

※ 다음 빈칸에 들어갈 단어를 채우시오.

1. 클래스 이름이 Rabbit일 때, 객체를 생성하기 위해서 다음과 같은 코드를 사용한다.

```
rabbit = 
```

2. 객체 이름이 rabbit일 때, 객체의 shape 속성에 "토끼"를 대입하기 위해서 다음과 같은 코드를 사용한다.

```
 = "토끼"
```

3. 객체 이름이 rabbit일 때, 객체의 goto() 행동에 100과 200을 대입하기 위해서 다음과 같은 코드를 사용한다.

```
 (100, 200)
```

정답

Click!

LAB 깡총깡총 뛰어다니는 토끼

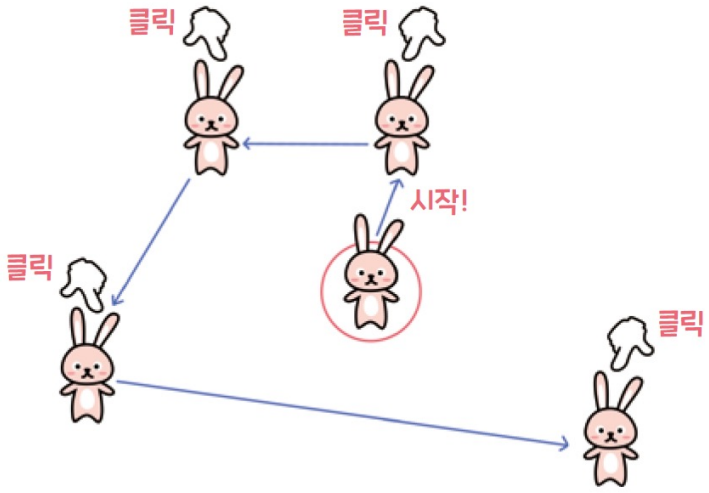
토끼 객체를 생성하고 사용자가 지정한 위치로 토끼가 이동하는 프로그램을 작성합시다.
거북이 그래픽과 달리 윈도우 창이 나오지 않기 때문에, 이동된 직후에 토끼의 위치를 텍스트로 출력해 봅시다.

실행 결과

```
토끼가 이동할  X좌표 ==>100
토끼가 이동할  Y좌표 ==>100
** 토끼의 현재 위치는 ( 100 , 100 )
토끼가 이동할  X좌표 ==>200
토끼가 이동할  Y좌표 ==>300
** 토끼의 현재 위치는 ( 200 , 300 )
토끼가 이동할  X좌표 ==> Ctrl+C를 눌러서 종료
```

사용자 입력

사용자 입력



LAB 깡총깡총 뛰어다니는 토끼

1. lab10-01.py 파일을 만들고, [코드 10-1]을 기반으로 Rabbit 클래스를 생성하기

```
## 클래스 및 함수 선언부
class Rabbit :
    shape = "" # 토끼 모양
    xPos = 0   # X위치
    yPos = 0   # Y위치

    def goto(self, x, y) :
        self.xPos = x
        self.yPos = y
```

2. 토끼 객체에 필요한 전역변수를 선언하기

```
## 전역변수 선언부
rabbit = None
userX, userY = 0, 0
```

LAB 깡총깡총 뛰어다니는 토끼

3. 토끼 객체를 생성하고 토끼 모양으로 바꾸기

```
## 메인 코드
rabbit = Rabbit()
rabbit.shape = "토끼"
```

4. 토끼가 이동할 x, y 좌표를 입력받고 토끼를 이동시키기

- 이동할 토끼의 위치를 출력하기

```
while True :
    userX = int(input("토끼가 이동할 X좌표 ==>"))
    userY = int(input("토끼가 이동할 Y좌표 ==>"))
    rabbit.goto(userX, userY)
    print("**토끼의 현재 위치는 (", str(userX), ",", str(userY),
")")
```

5. <Ctrl>+<S>를 눌러서 변경된 내용을 저장하고, <F5>를 눌러 실행 결과 확인하기

Section 03
생성자와 특별한 메소드

생성자의 개념

- 생성자
  - 객체를 생성하면 무조건 호출되는 메소드를 의미함
  - 객체를 생성하면서 변수의 값을 초기화하는 메소드
  - 생성자로 초기화를 하면 코드가 간결해짐

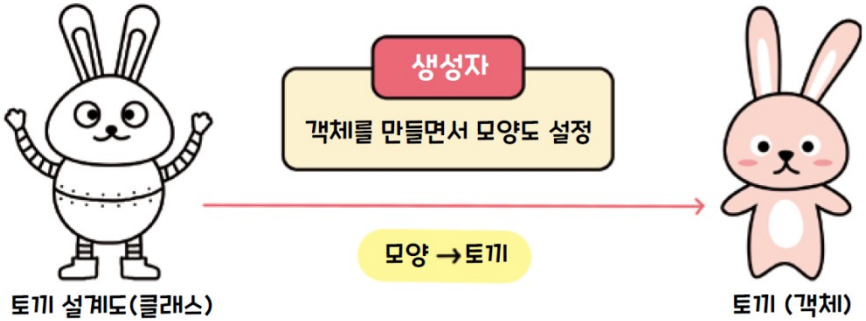


그림 10-9 생성자의 개념

생성자의 개념

생성자의 형태

- 생성자는 클래스 안에서 __init__()라는 이름으로 지정되어 있음

```
class 클래스이름 :
    def __init__(self) :
        # 초기화할 코드 입력
```

__init__()는 앞뒤에 언더바(_)가 2개씩 있습니다. init은 Initialize의 약자로 초기화 한다는 의미를 갖습니다.

생성자의 개념

생성자의 형태

[코드 10-2]

```
class Rabbit :
    shape = "" # 토끼 모양
    xPos = 0   # X위치
    yPos = 0   # Y위치

    def __init__(self) :
        self.shape = "토끼"

    def goto(self, x, y) :
        self.xPos = x
        self.yPos = y

rabbit = Rabbit() # 토끼 객체 생성
print('rabbit의 모양:', rabbit.shape)
```

[실행결과]

```
rabbit의 모양: 토끼
```

생성자의 개념

- 매개변수가 있는 생성자
  - 생성자도 다른 메소드처럼 매개변수(=파라미터)를 사용할 수 있음

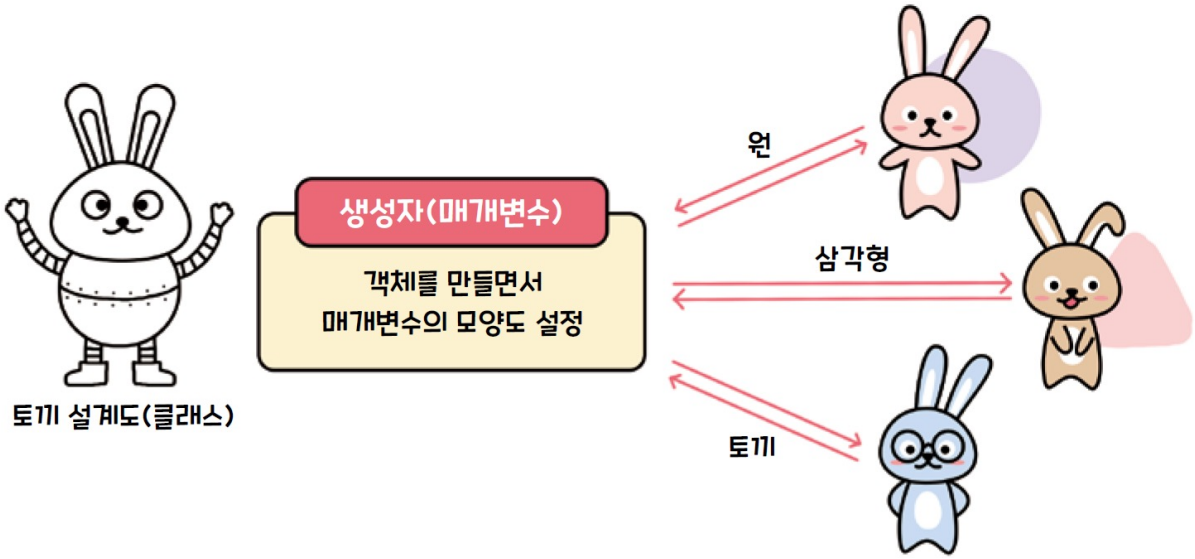


그림 10-10 매개변수가 있는 생성자의 개념

생성자의 개념

- 매개변수가 있는 생성자

[코드 10-3]

```
class Rabbit :
    shape = "" # 토끼 모양
    xPos = 0   # X위치
    yPos = 0   # Y위치

    def __init__(self, value) :
        self.shape = value

    def goto(self, x, y) :
        self.xPos = x
        self.yPos = y

rabbit1 = Rabbit('원')
print('rabbit1의 모양:', rabbit1.shape)

rabbit2 = Rabbit('삼각형')
print('rabbit2의 모양:', rabbit2.shape)

rabbit3 = Rabbit('토끼')
print('rabbit3의 모양:', rabbit3.shape)
```

[실행결과]

```
rabbit1의 모양: 원
rabbit2의 모양: 삼각형
rabbit3의 모양: 토끼
```

생성자의 개념

확인문제

※ 다음 빈칸에 들어갈 단어를 채우시오.

1. 객체를 생성하면서 속성의 값을 초기화시키는 메소드를 [ ](이)라 한다.

2. 생성자는 파이썬 안에서 이름이 [ ](으)로 지정되어 있다.

3. 생성자도 매개변수를 받을 수 있습니다. 다음 코드는 생성자가 매개변수를 전달받아서 속성 중 shape에 대입하는 코드이다.

```
def __init__(self, value) :
    self.shape = [ ]
```

정답

Click!

특별한 메소드

- __del__( ) 메소드
  - __init__() 메소드가 생성자라면, __del__() 메소드는 소멸자라고 부름
  - __del( )__는 객체가 제거될 때 자동으로 호출됨
    - 객체를 제거할 때는 del(객체)로 지우는데, 이때 호출됨

[코드 10-4]

```
class Rabbit :
    shape = ""
    def __del__(self) :
        print("이제", self.shape, "는 자유에요~~")

rabbit = Rabbit()
rabbit.shape = "도비"
del(rabbit)
```

[실행결과]

```
이제 도비는 자유에요~~
```

특별한 메소드

- __add__( ) 메소드
  - 객체끼리 덧셈을 할 경우에 실행되는 메소드
  - 일반적으로 덧셈(+) 연산은 숫자나 문자열 등에만 작동하지만 __add__() 메소드를 작성해 놓으면 객체 사이의 덧셈 작업도 가능함

[코드 10-5]

```
class Rabbit :
    shape = ""
    def __add__(self, other) :
        print("객체", self.shape,"와", other.shape, "가 친구가
되었습니다.")

rabbit1 = Rabbit()
rabbit1.shape = "엽기토끼"
rabbit2 = Rabbit()
rabbit2.shape = "도비"

rabbit1 + rabbit2
```

[실행결과]

```
객체 엽기토끼와 도비가 친구가 되었습니다.
```

특별한 메소드

하나 더 알기 **비교 메소드**

객체 사이의 비교 연산자가 사용될 때 호출되는 메소드도 있습니다. 사용법은 바로 다음 [LAB]에서 확인해 봅니다.

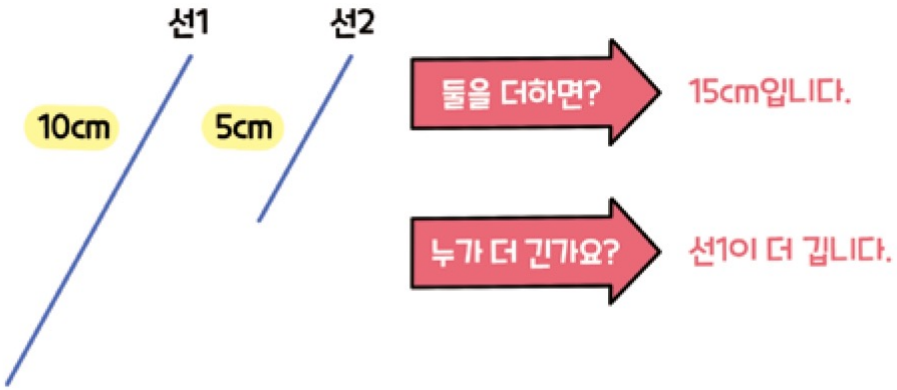
표 10-1 객체 사이의 비교 메소드

비교 연산자	메소드
<	__lt__()
<=	__le__()
>	__gt__()
>=	__ge__()
==	__eq__()
!=	__ne__()

LAB 선을 객체로 활용하기

2개의 선을 더하고 비교하는 프로그램을 만들어 봅시다.

그런데 파이썬에는 선(Line)이라는 데이터 형식은 없기 때문에 선이라는 객체를 만들어서 사용하려고 합니다. 우리가 만들 선의 속성으로는 길이를 지정하겠습니다. 메소드로는 서로 다른 2개 선을 더하면 길이가 합해지는 기능과 2개 선의 크기를 비교하는 기능을 만듭니다. 비교해서 어느 것이 더 긴지 알려주고, 짧은 선은 삭제하는 기능을 합니다.



실행 결과

```
10 길이의 선이 생성되었습니다.
5 길이의 선이 생성되었습니다.
두 선의 합 :  15
선1이 더 깁니다.
5 길이의 선이 제거되었습니다.
```

LAB 선을 객체로 활용하기

1. lab10-02.py 파일을 만들고, 선(Line) 클래스를 생성하기. 우선 선 클래스에서 생성자를 만들기

```
## 클래스 선언
class Line :
    length = 0

    def __init__(self, length) :
        self.length = length
        print(self.length, '길이의 선이 생성되었습니다.')
```

2. 선을 삭제할 del(객체) 코드를 만나면 실행되는 소멸자 __del__() 메소드를 생성하기

```
    def __del__(self) :
        print(self.length, '길이의 선이 제거되었습니다.')
```

3. 두 선의 객체를 더할 때 실행되는 __add__() 메소드를 만들기

```
    def __add__(self, other) :
        return self.length + other.length
```

LAB 선을 객체로 활용하기

4. __lt__()는 두 선의 객체를 작다(<) 연산자로 비교할 때 실행됨
   - self.length가 짧으면 True를, other.length가 짧으면 False를 반환함

```
    def __lt__(self, other) :
        return self.length < other.length
```

5. __eq__() 메소드는 두 선의 길이를 비교함
   - 실행됩니다. self.length가 같으면 True를, other.length가 다르면 False를 반환함

```
    def __eq__(self, other) :
        return self.length == other.length
```

6. 2개의 선 객체를 만들기

```
## 메인 코드 부분
line1 = Line(10)
line2 = Line(5)
```

LAB 선을 객체로 활용하기

7. 2개 선의 길이 합을 구하기 위해 객체의 합계를 출력하기

```
print('두 선의 합 : ', line1 + line2)
```

8. 두 선을 비교하기 위해 __lt__() 및 __eq__() 메소드를 호출함
   - 이때 두 선을 비교하여 더 짧은 선은 삭제함

```
if line1 < line2 :
    print('선2가 더 깁니다.')
    del(line1)
elif line1 == line2 :
    print('두 선의 길이가 같습니다.')
else :
    print('선1이 더 깁니다.')
    del(line2)
```

9. <Ctrl>+<S>를 눌러서 변경된 내용을 저장하고, <F5>를 눌러 실행 결과 확인하기

Section 04
클래스와 상속

상속의 개념

- 상속
  - 기존의 클래스가 가지고 있는 속성과 행동을 그대로 물려받는 새로운 클래스를 만드는 것을 의미함
    - 자식에게 유산을 물려주는 개념과 같은 개념
  - 상속을 받은 이후에는 새로운 클래스에서 추가로 속성이나 행동을 만들어서 사용할 수도 있음
- 상속의 예시 : 집토끼 클래스와 산토끼 클래스

그림 10-11 집토끼와 산토끼 클래스 개념

상속의 개념

- 상속의 예시 : 집토끼 클래스와 산토끼 클래스
  - 집토끼와 산토끼의 공통된 특징을 토끼라는 상위 클래스에 갖도록 하기
  - 슈퍼 클래스 : 상위 클래스인 토끼 클래스
  - 서브 클래스 : 하위 클래스인 집토끼와 산토끼 클래스

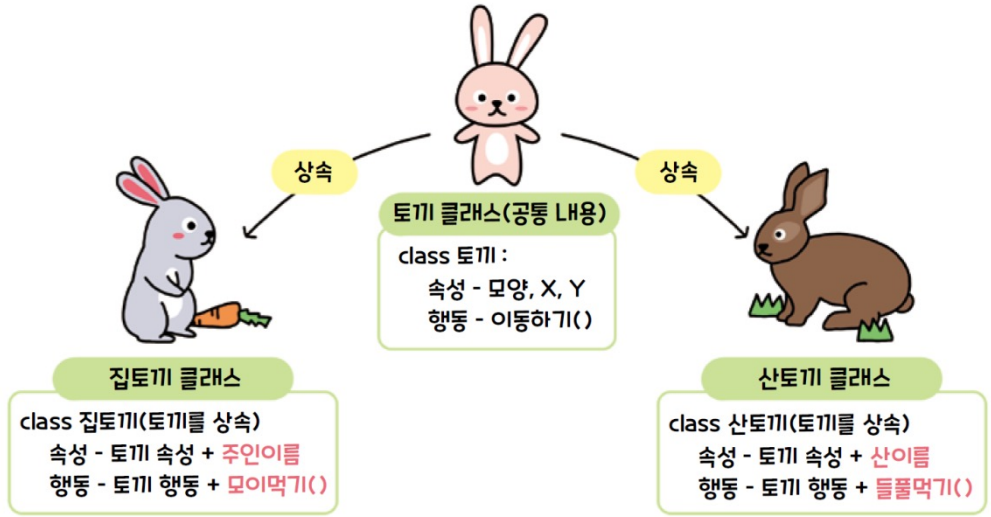


그림 10-12 상속의 개념

상속의 구현

- 상속 구현하기
  - 서브 클래스를 정의할 때, 괄호 안에 슈퍼 클래스의 이름을 넣어줌

```
class 서브 클래스(슈퍼 클래스) :
    # 서브 클래스 코드 구현
```

  - 슈퍼 클래스인 토끼 클래스(Rabbit)와 서브 클래스인 집토끼(HouseRabbit), 산토끼(Mountain Rabbit) 클래스를 구현하기

상속의 구현

- 상속 구현하기

[코드 10-6]

```
class Rabbit :
    shape = ""
    xPos = 0
    yPos = 0
    def goto(self, x, y) :
        self.xPos = x
        self.yPos = y

class HouseRabbit(Rabbit) :
    owner = ""
    def eatFeed() :
        print("집토끼가 모이를 먹습니다")

class MountainRabbit(Rabbit) :
    mountain = ""
    def eatWildglass() :
        print("산토끼가 들풀을 먹습니다")
```

상속의 구현

- 상속 구현하기
  - 서브 클래스의 객체를 생성

```
hRabbit = HouseRabbit()
mRabbit = MountainRabbit()
```

  - 두 객체 사용하기

```
hRabbit.shape = '원'
mRabbit.goto(100,100)
```

상속의 구현

확인문제

1. 다음 중 잘못된 것을 고르시오.

   ① 부모 클래스로부터 상속받으면, 더 이상 속성이나 메소드를 추가할 수 없다.

   ② 객체끼리도 상속하거나 받을 수 있다.

   ③ 자식 클래스에는 없는 속성이더라도 부모 클래스에 있는 속성이면 사용할 수 있다.

   ④ 부모 클래스에 없는 메소드라도 자식 클래스에서 메소드를 추가할 수 있다.

2. 슈퍼 클래스인 학생(Student) 클래스의 상속을 받는 대학생(CollegeStudent) 클래스와 관련된 코드이다. 다음 빈칸에 들어갈 단어를 채우시오.

```
class CollegeStudent (          ) :
    // 이곳에 코딩
```

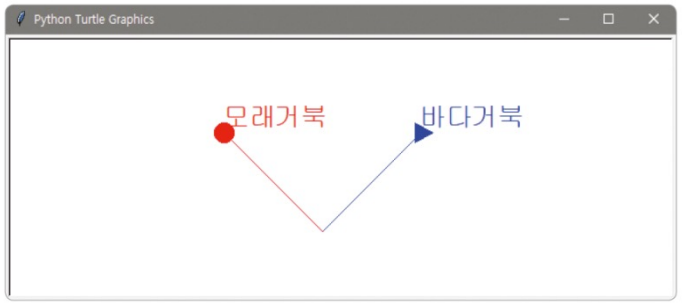
정답

Click!

LAB 바다거북과 모래거북

기존 파이썬에서 제공하는 거북이를 슈퍼 클래스로 지정할 수도 있습니다.

거북이를 상속받아서 바다거북 및 모래거북을 생성해 봅시다. 그리고, 바다거북은 수영하는 메소드를, 모래거북은 걸어다니는 메소드를 따로 만들겠습니다.



LAB 바다거북과 모래거북

1. lab10-03.py 파일을 만들고, 거북이를 상속받는 바다거북 클래스를 생성하기
   - 생성자에서 바다거북의 모양, 색상 및 이름을 지정하기
   - 바다거북이므로 swim() 메소드를 추가하기
     - 메소드의 내용은 goto()로 거북이가 해당 좌표로 이동함

```
import turtle
## 클래스 및 함수 선언부
class SeaTurtle(turtle.Turtle) :
    name = ''
    body = None
    def __init__(self) :
        self.body = turtle.Turtle('triangle')
        self.body.color("blue")
        self.name = "바다거북"
    def swim(self, x, y) :
        self.body.goto(x, y)
```

LAB 바다거북과 모래거북

2. 같은 방식으로 모래거북 클래스를 생성하기
   - 생성자에서 모래거북의 모양, 색상, 이름을 바다거북과 다르게 지정함
   - 모래거북은 walk() 메소드를 추가함

```
class SandTurtle(turtle.Turtle) :
    name = ''
    body = None
    def __init__(self) :
        self.body = turtle.Turtle('circle')
        self.body.color("red")
        self.name = "모래 거북"
    def walk(self, x, y) :
        self.body.goto(x, y)
```

3. 바다거북 및 모래거북 객체에 필요한 전역변수를 선언하기

```
## 전역변수 선언부
seaTut, sandTut = None, None
```

LAB 바다거북과 모래거북

4. 바다거북 및 모래거북 객체를 생성하기
   - 이때 각 거북의 생성자가 실행됨

```
# 메인 코드
seaTut = SeaTurtle()
sandTut = SandTurtle()
```

5. 바다거북을 지정한 위치까지 수영을 시키고, 자신의 이름을 이동한 위치에 쓰도록 함
   모래거북은 지정한 위치까지 걷고, 역시 자신의 이름을 쓰도록 함

```
seaTut.swim(100, 100)
seaTut.body.write(seaTut.name, font=("Arial", 20))

sandTut.walk(-100, 100)
sandTut.body.write(sandTut.name, font=("Arial", 20))
```

6. <Ctrl>+<S>를 눌러서 변경된 내용을 저장하고, <F5>를 눌러 실행 결과 확인하기

[실전 예제]
GPS를 단 토끼

실전 예제 GPS를 단 토끼

[문제]

- 거북이가 토끼를 등에 업고 바닷속을 유람시켜주자
- 이때 GPS를 단 토끼는 거북이가 이동할 때마다 현재 자신의 위치를 알려준다
    - 거북이 등 위의 토끼 위치를 알려주는 프로그램을 작성하기 위해서는 거북이 객체와 토끼 객체를 하나로 묶는 방법을 사용함

실행 결과

```
** 토끼가 거북이 등에 올라탔습니다. **
거북이 등 위의 토끼 위치는 현재 -183, 92입니다.
거북이 등 위의 토끼 위치는 현재 49, -216입니다.
거북이 등 위의 토끼 위치는 현재 9, 246입니다.
거북이 등 위의 토끼 위치는 현재 220, 232입니다.
거북이 등 위의 토끼 위치는 현재 232, 56입니다.
```

실전 예제 GPS를 단 토끼

[해결]

```
import turtle
import random

## 클래스 선언부
class Rabbit :
    myTurtle = None
    def __init__(self, myTut) :
        self.myTurtle = myTut
        print("** 토끼가 거북이 등에 올라탔습니다. **")
    def print_my_position(self) :
        print("거북이 등 위의 토끼 위치는 현재",
              self.myTurtle.xcor( ),","self.myTurtle.ycor(),"입니다.")

## 전역변수 선언부
myTut, myRab = None, None
colorList = ["red", "green", "blue", "black", "magenta", "orange", "gray"]

## 메인 코드부
turtle.setup(550, 550)
turtle.screensize(500, 500)

myTut = turtle.Turtle("turtle")
myRab = Rabbit(myTut)
myTut.pensize(5)
```

실전 예제 GPS를 단 토끼

[해결]

```
for _ in range(20) :
    x = random.randint(-250, 250)
    y = random.randint(-250, 250)
    color = random.choice(colorList)
    myTut.pencolor(color)
    myTut.goto(x,y)
    myRab.print_my_position( )

turtle.done( )
```

Preview

감사합니다 :)