파이썬 객체지향 기초

객체란?

속성과 기능을 포함한 프로그램의 단위로 구성요소를 객체화하는 것을 OOP라고 한다.

Class Bike:

#속성은 init 안에 넣는다.

Def \_\_init\_\_(self): #생성자함수 : 객체가 생성될 때 가장 먼저 호출되는 함수

Self.color=’black’

Self.weight=3

Def drive(self):

Print(‘drive’)

Def brake(self):

Print(‘brake’)

Def gear(self):

Print(‘gear’)

클래스란? 객체를 생성하기 위한 틀

접근자 (public, private)

Class PayGildong:

Def \_\_init\_\_(self):

Self.day=25 #public 설정

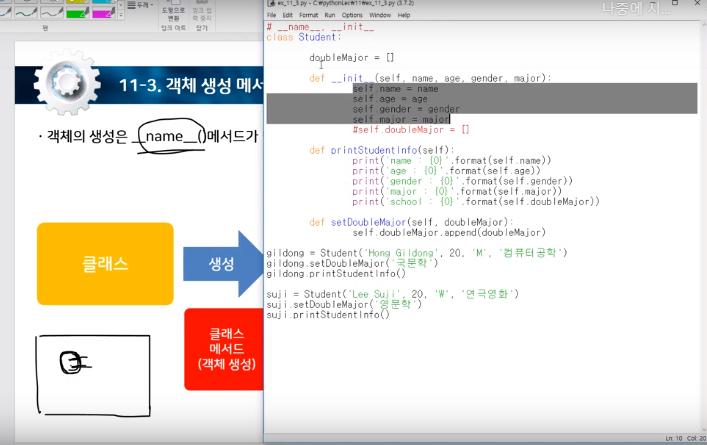
Self.\_\_money = 100000 #private 설정

# 파이썬 메서드의 첫 번째 인자로 항상 인스턴스가 전달된다.

객체의 생성은 \_\_name\_\_() 메서드가 담당하고, 초기화는 \_\_init\_\_() 메서드가 담당한다.

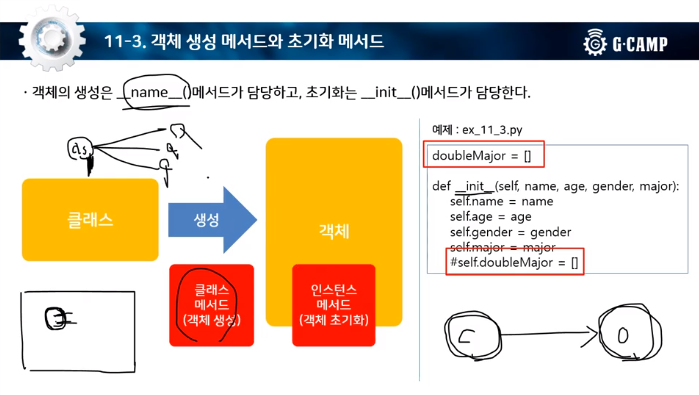
메모리에 비어있는 객체를 탑재해 주는 역할을 \_\_name\_\_ 이 담당한다.

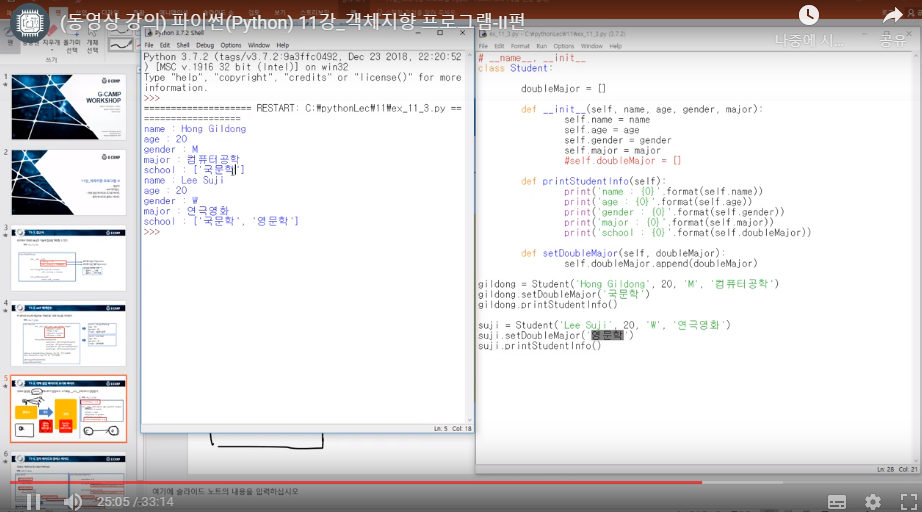
비어있는 객체에 속성과 기능을 초기화 해주는 것이 \_\_init\_\_이 한다.



객체가 생성되고 난 후에 생성되는 인스턴스 메서드

doubleMajor는 클래스 메서드. 객체(인스턴스)가 생성되기 전에 클레스에서 만들어진 부분



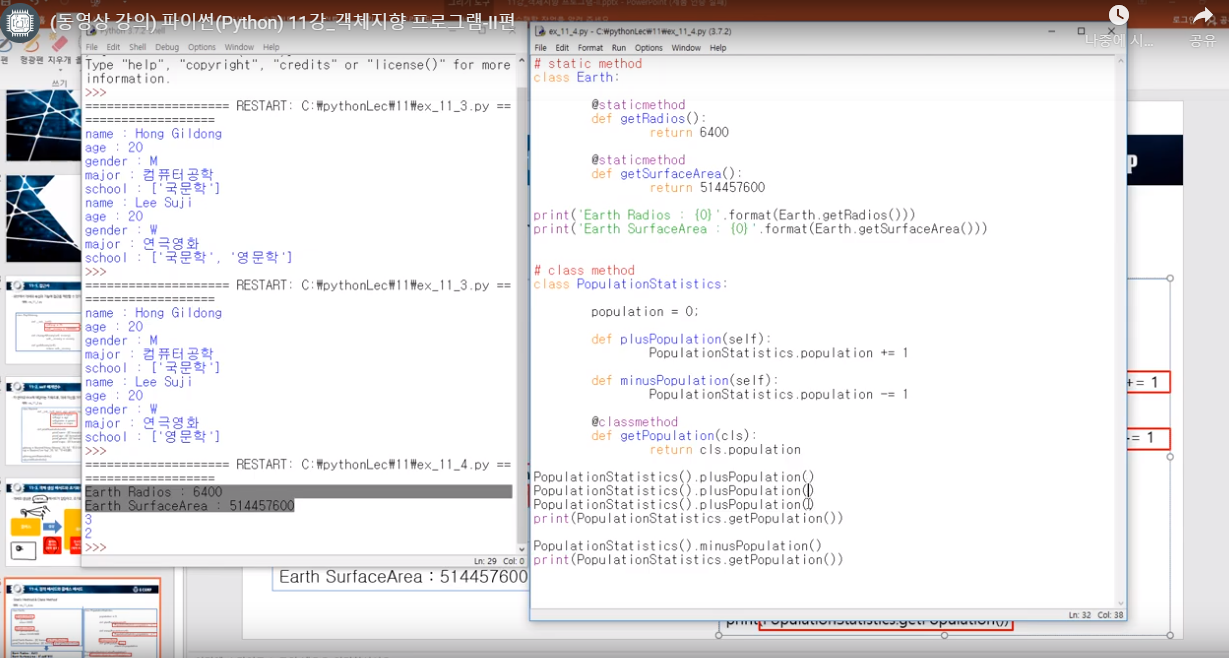


doubleMajor는 클래스 속성이기에, 모든 객체에 공유가 되는 속성이기에 누적되어 append 된것을 볼 수 있다. (국문학, 영문학)

<https://www.inflearn.com/course/%ED%8C%8C%EC%9D%B4%EC%8D%AC-%ED%94%84%EB%A1%9C%EA%B7%B8%EB%9E%98%EB%B0%8D/lecture/17889>

인스턴스 메소드, 클래스 메소드, 스태틱 메소드 이 세 가지 메소드의 개념을 확실히 잡고 가도록 하죠. 이 세 가지 메소드는 모두 클래스 안에서 정의 됩니다. 인스턴스 메소드는 인스턴스를 통해서 호출이 되고, 첫 번째 인자로 인스턴스 자신을 자동으로 전달합니다. 관습적으로 이 인수를 'self'라고 칭합니다. 클래스 메소드는 클래스를 통해서 호출이 되고 "@classmethod"라는 데코레이터로 정의합니다. 첫 번째 인자로는 클래스 자신이 자동으로 전달되고 이 인수를 관습적으로 'cls'라고 칭합니다. 스태틱 메소드는 앞서 설명한 두 메소드와는 틀리게 인스턴스나 클래스를 첫 번째 인자로 받지 않습니다. 스태틱 메소드는 클래스 안에서 정의되어 클래스 네임스페이스 안에는 있을뿐 일반 함수와 전혀 다를게 없습니다. 하지만 클래스와 연관성이 있는 함수를 클래스 안에 정의하여 클래스나 인스턴스를 통해서 호출하여 조금 편하게 쓸 수가 있는 것 입니다. 밑의 예를 보시죠.

<http://schoolofweb.net/blog/posts/%ED%8C%8C%EC%9D%B4%EC%8D%AC-oop-part-4-%ED%81%B4%EB%9E%98%EC%8A%A4-%EB%A9%94%EC%86%8C%EB%93%9C%EC%99%80-%EC%8A%A4%ED%83%9C%ED%8B%B1-%EB%A9%94%EC%86%8C%EB%93%9C-class-method-and-static-method/>

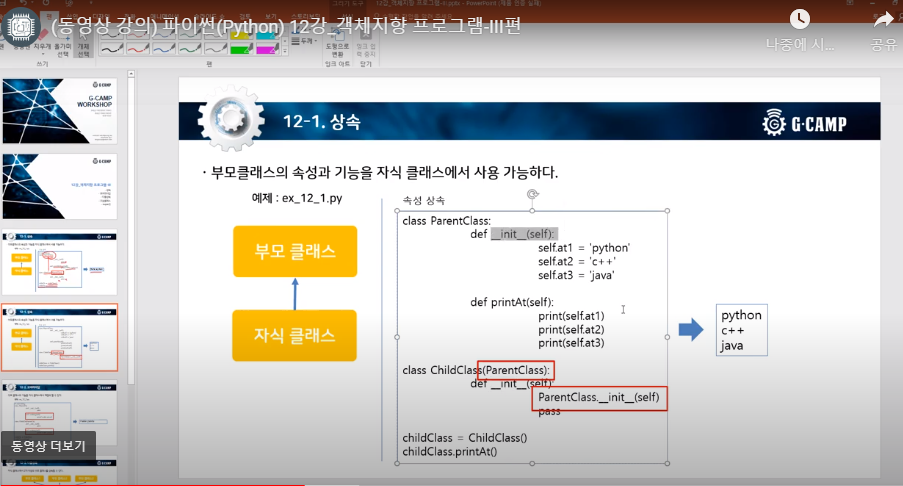


클래스메서드를 호출할 때는 ‘클래스명.클래스메서드()’로 호출하여야 한다.

클래스메서드의 첫번째 파라미터에는 클래스 자신이 자동으로 전달되고 이 인수를 관습적으로 ‘cls’ 라고 한다고 한다.

## 객체지향 프로그램 3

### 상속



부모 클래스를 상속해서는 public은 가져올 수 있지만 private은 가져올 수 없다.

또한 부모 클래스의 속성을 가져올려면 자식 클래스에서 ParentClass.\_\_init\_\_(self) 를 실행하여야 한다.

왜냐하면 childClass를 생성될 때 가장 먼저 실행되는 메소드는 child 클래스의 \_\_init\_\_ 메서드이다. (\_\_init\_\_ 메소드 안에서 속성값들이 메모리에 다 탑제가 된다.)

이 과정에서 부모 클래스의 \_\_init\_\_() 메서드는 호출이 되지 않는다.

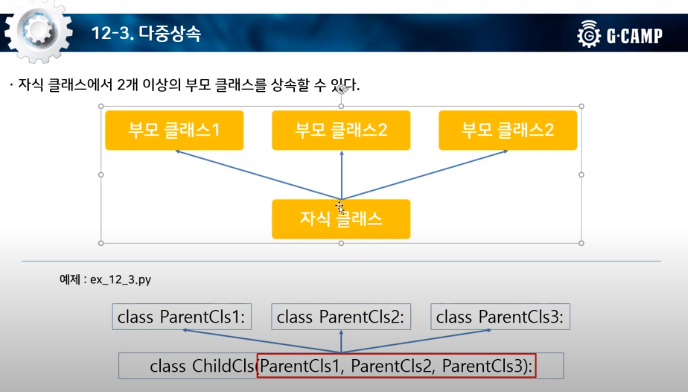
Child클래스를 생성했기 때문에 메모리에서는 child 클래스의 init()메서드만 호출한다. 상속받은 부모 클래스의 \_\_init\_\_ 메서드는 호출하지 않는다.

그렇기에 ‘부모클래스.\_\_init\_\_(self)’ 처럼 부모클래스의 생성자를 수동으로 호출하여 부모의 속성 값을 초기화 할 수 있다.

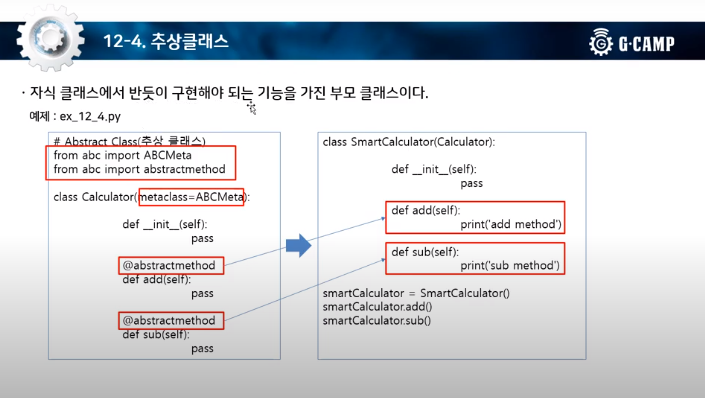
### 오버라이딩

부모 클래스의 기능을 자식 클래스에서 재정의 할 수 있다.

### 다중상속



### 추상클래스



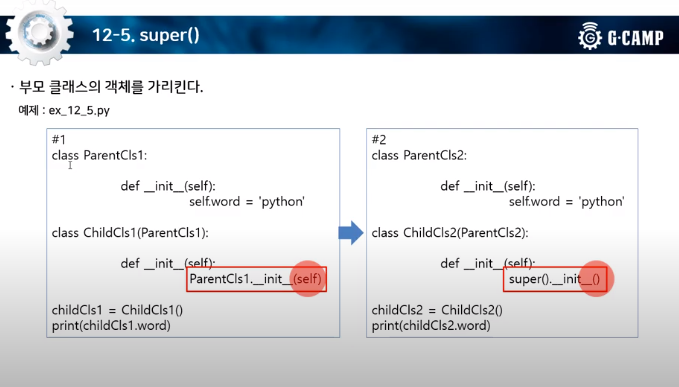
추상클래스를 사용하기 위해서는 위 package에서 2개의 module을 import 해야 한다.

클래스 생성할 때 (metaclass=ABCMeta) 를 넣어서 추상클래스임을 알려야 한다.

추상메서드 위에 @abstractmethod 임을 알리는 데코레이터를 붙여서 추상클래스 최종 완성

### Super()

부모 클래스의 객체를 가리킨다.



### 다중상속시 super() 동작원리?

<https://uzooin.tistory.com/137?category=739703>

다중 상속의 경우 super()를 이용하면 첫번째 부모만을 의미한다.

그렇기에 양쪽 부모의 생성자를 호출하기 위해서는 부모 클래스의 이름을 이용해서 직접 생성자를 호출해야 한다.

Ex) 부모클래스명.\_\_init\_\_(self)

부모클래스명2.\_\_init\_\_(self)

또한 부모의 클래스에 같은 이름의 메소드가 존재한다면?

자식클래스.메소드명() 호출하였을 때, 첫번째 부모 클래스에서부터 해당 이름의 메소드가 존재하는지 찾아서 먼저 찾은 메소드를 호출한다.

부모 클래스에 있는 같은 이름의 메서드를 자식클래스에 둘 다 사용하기 위해서는 아래 소스처럼 자식클래스에서 ‘부모클래스.동일한이름의메서드명()’ 을 호출하는 메서드를   
따로 정의해서 사용해야 한다고 한다.

class ParentOne:

    def \_\_init\_\_(self,one):

        self.\_one = one

    def func(self):

        print('call ParentOne.func')

class ParentTwo:

    def \_\_init\_\_(self,two):

        self.\_two = two

    def func(self):

        print('call ParentTwo.func')

class Child(ParentOne, ParentTwo):

    def \_\_init\_\_(self, one, two, three):

        ParentOne.\_\_init\_\_(self,one)

        ParentTwo.\_\_init\_\_(self,two)

        self.\_three = three

    def oneFunc(self):

        ParentOne.func(self)

    def twoFunc(self):

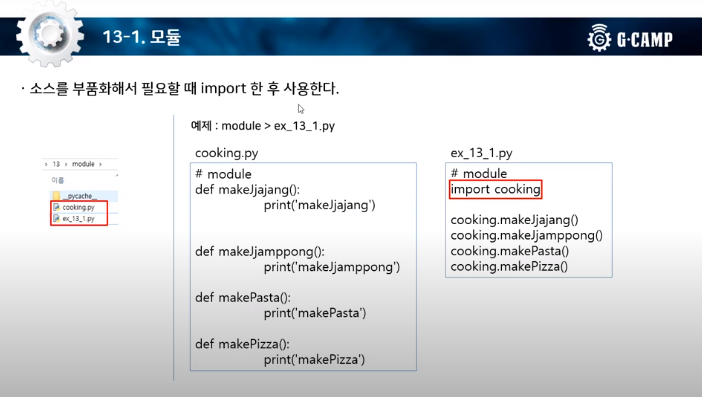
        ParentTwo.func(self)

child = Child(1,2,3)

child.oneFunc()

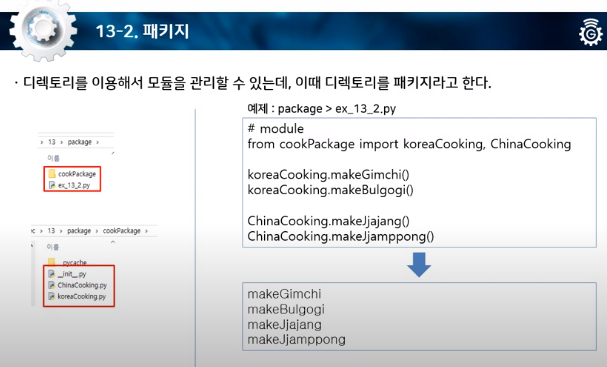
child.twoFunc()

## 모듈과 패키지



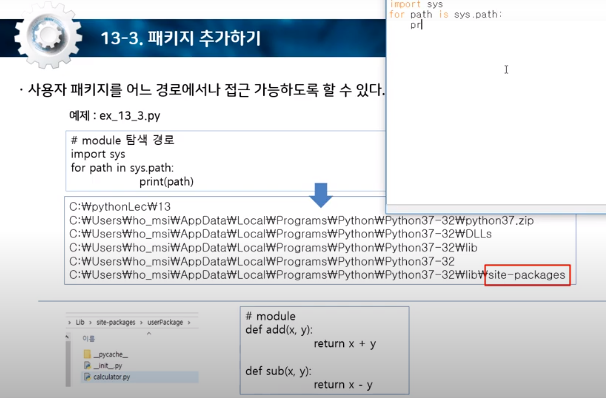
Import 파일명(.py 땐것) 으로 해당 파일안에 들어있는 모든 함수를 가져올 수 있다.

From 파일명 import 모듈명 으로 해당 파일 안에서 내가 필요한 모듈만 가져올 수 도 있다.



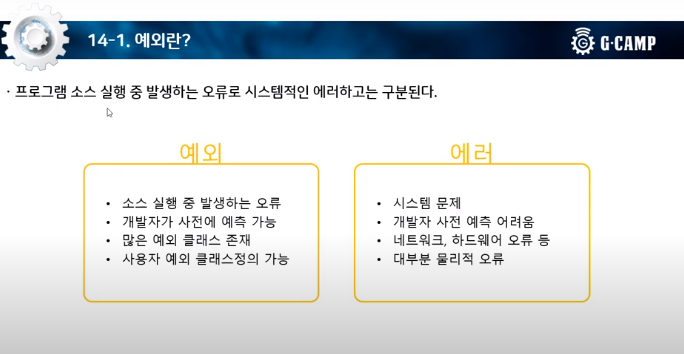
From folder import file, file1

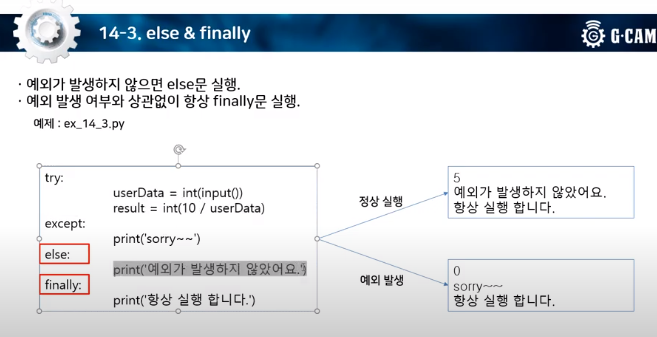
패키지 폴더를 만들어서 쓰기 위해서는 패키지 폴더 안에 모듈.py 파일들 말고도 \_\_init\_\_.py(내용 없어도 된다) 파일을 생성하여야 한다.

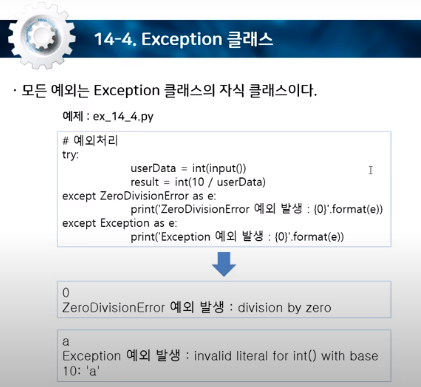


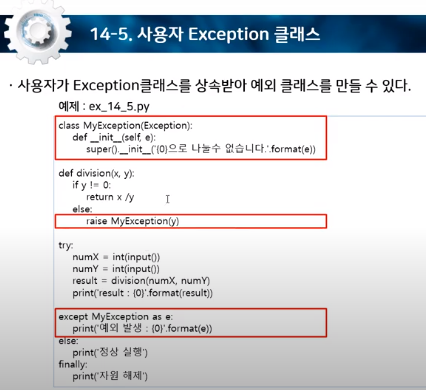
자신이 만들어 놓은 패키지를 위에 표시되어있는 site-package 폴더 아래에 두면, 파이썬 인터프리터가 모듈을 찾을 때 site-package 폴더 아래를 찾으므로 패키지를 경로 없이 넣어서 사용할 수 있다.

## 예외처리





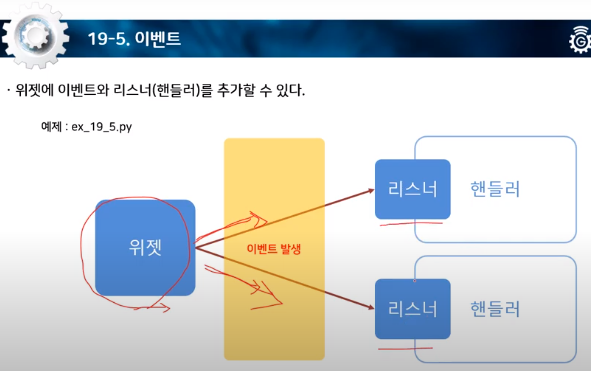




파이썬 인터프리터에서 기존에는 알아서 exception을 raise 해줬는대 이 경우는 사용자가 직접 예외를 정의하고, 예외 던질 상황을 지정한다음 쓰는 상황이다.

## 19 UI 구성하기

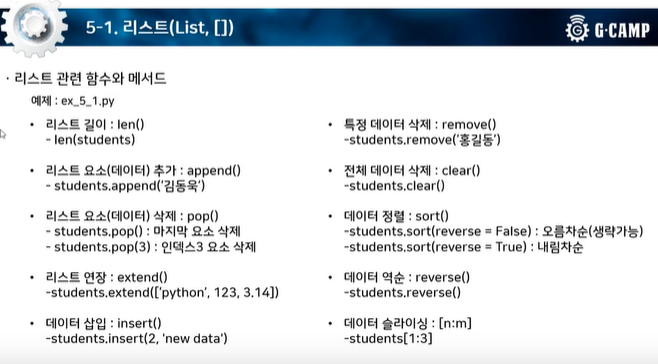
이벤트

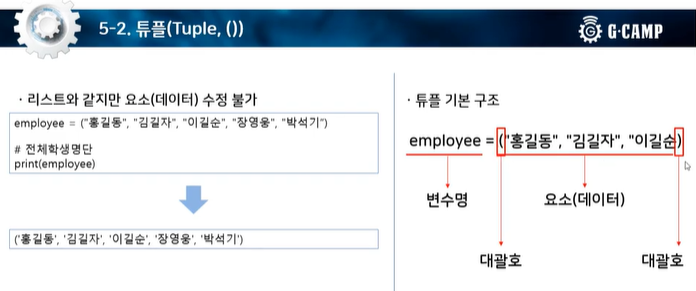


이벤트 발생시 브로드캐스팅

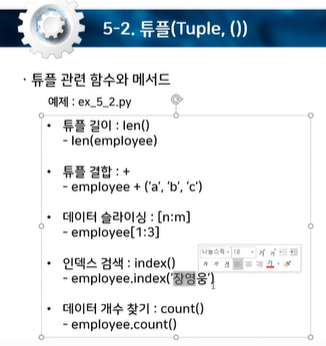
이벤트 시그널을 수신할 리스너와 헨들러를 만들어야 한다.

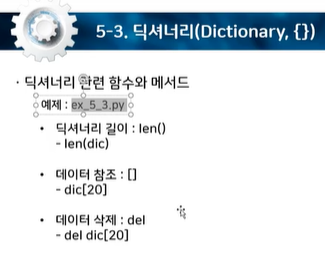
5. 컨테이너 자료형





튜플 – 리스트와 같지만 수정 불가





8. 반복문

