클래스 정의

객체에 대한 설계도 : 클래스

클래스로부터 만들어지는 각각의 객체를 그 클래스의 인스턴스 라고 함.

Class ClassName:

Stmt-1…

이렇게 함.

클래스의 몸체에 pass만 있는 경우

class Emp:

pass

x = Emp()

y = Emp()

print(x==y)

* 결과는 다르다. 클래스에서는 바탕은 똑같지만 x와 y로 객체의 이름이 다르므로 false이다.

존재하는 객체에서 동적으로 속성 생성이 가능하다.

class Emp:

pass

x = Emp()

y = Emp()

x.name = "buseok"

x.age = 23

print(x.name,x.age)

print(x.\_\_dict\_\_)

print(y.\_\_dict\_\_)

* buseok 23
* {'name': 'buseok', 'age': 23}
* {}

객체의 동적 할당

class Emp:

pass

x = Emp()

y = Emp()

#x와 y가 아무것도 할당되지 않았을 때

Emp.dept = 'reserch'

print(x.dept,y.dept)

#x에 동적으로 속성 생성

x.dept = 'fiqjw'

print(x.dept,y.dept)

#다시 dept에 값을 주었을 때

x.dept = '12ㅅㅈㅂㄹ'

print(x.dept,y.dept)

객체 속성 연결

함수도 function clss

함수명에도 속성 설정

def f(x):

return 42

f.x = 47

print(f(1))

* 42

def f(x):

return 42

f.x = 47

print(f.x)

* 47

객체를 인수로 하는 함수 intro()정의

def intro(emp):

print(emp.name)//emp = 단순변수

class Emp://Emp = 클래스

pass

a = Emp()

a.name = "sexking"

intro(a)

* Sexking

함수 intro를 클래스 속성인 myintro에 연결

def intro(emp):

print(emp.name)

class Emp:

myintro = intro

a = Emp()

a.name = "sexking"

Emp.myintro(a)

* Sexking

메소드 정의 및 활용

class Emp:

def restart(self):

self.bonus = 0

def sale(self):

self.bonus += 10

a = Emp()

a.name = 'asd'

a.restart()

a.sale()

print(a.name + '의 보너스는', a.bonus)

* Asd의 보너스는 10

클래스 : 예제

class cnt:

def reset(self):

self.count = 0

def inc(self):

self.count += 1

def get(self):

return self.count

a= cnt()

a.reset()

a.inc()

print(a.get())

* 1이 호출됨.

생성자

메소드 : \_\_init\_\_()

class Emp:

empt = 0

def \_\_init\_\_(self,name,bonus=100):

self.name = name

self.bonus = self.base = bonus

Emp.empt += 1

def res(self):

self.bonus = self.base

def sal(self):

self.bonus += 10

a = Emp('Peter', 50)

b = Emp('ase')

a.sal()

print(a.name+'의 보너스;', a.bonus)

print(b.name+'의 보너스;', b.bonus)

print(Emp.empt)

\_\_str\_\_() 메소드

객체를 print() 로 출력할 때,자동적으로 호출

정보 은닉

변수 앞에 \_\_뭍임.

클래스 외부에서 private 변수를 사용하기 위해, 그 값을 반환하는 메소드를 제공해야 함.

접근자(getters) 인스턴스 변수값을 반환

설정자(setters) 인스턴스 변수값을 설정

class Emp:

empt = 0

def \_\_init\_\_(self,name,bonus=100):

self.\_\_name = name

self.\_\_bonus = self.base = bonus

Emp.empt += 1

def res(self):

self.\_\_bonus = self.base

def sal(self):

self.\_\_bonus += 10

def getName(self):

return self.\_\_name

def getBonus(self):

return self.\_\_bonus

a = Emp('Peter', 50)

b = Emp('ase')

a.sal()

print(a.getName()+'의 보너스;', a.getBonus())

print(b.getName()+'의 보너스;', b.getBonus())

print(Emp.empt)

* Getname,getbonus 가 접근자가 되어 정상적으로 실행됨.