

# LAB 10

2020-01 소프트웨어의 이해 01분반 / 조교 이경민

# LAB 10

---

- 객체
- 실습 과제

# 객체 (object)

---

- 함수와 변수를 하나의 단위로 묶을 수 있는 방법
- 쉽게 하나의 물건이라고 생각하기!
- Python = 객체 지향 언어
- 예시) 사람, TV... : 각각 특정한 기능을 수행하는 객체



# 객체 (object)

- 객체 = 속성(변수) + 동작(메소드)
- **속성(attribute)** : 객체가 가지는 변수라고 생각 (사용 방식은 일반 변수와 동일)  
ex) tv.brightness = 50
- **동작(action)** : 객체가 가지는 함수 기능이라고 생각  
ex) tv.changeBrightness(50)

속성
메이커
모델
색상
연식
가격

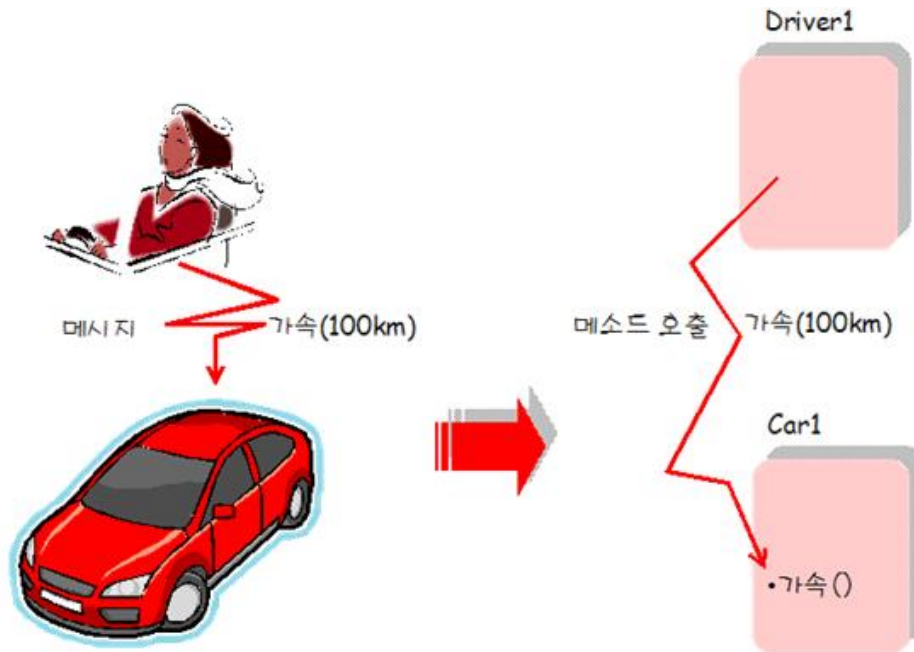


동작
주행하기
방향바꾸기
주차하기

사람
+ 이름
+ 생일
+ 성별
+ 직업
+ 주소
+ 일하다()
+ 공부하다()
+ 잠자다()
+ 밥먹다()
+ 말하다()

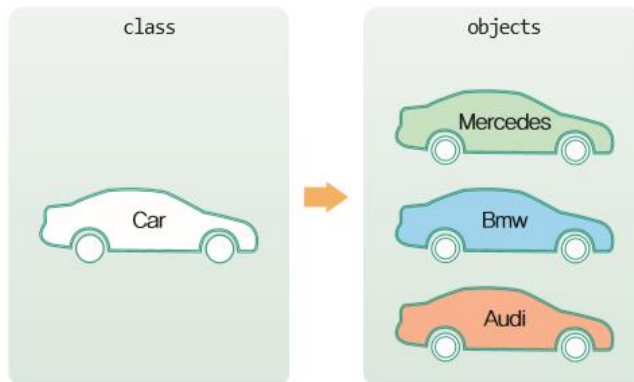
# 객체 (object)

- 객체는 메시지를 통해 서로 상호작용



# 객체 생성하는 방법

- 1) 객체의 설계도인 class를 작성
  - 2) 클래스로부터 객체를 생성
- 
- **클래스 (class)** : 설계도 및 틀이라 생각
  - **객체 (object)** : 클래스로부터 생성된 실체  
ex) 생산되는 구체적 모델이라고 생각



## 예제) 자동차 객체 생성

- Car 클래스 생성 후, myCar이라는 객체 생성
- 속성
  - speed, model, color, year
- 메소드
  - drive()

```
자동차 객체를 생성하였습니다.  
자동차의 속도는 0  
자동차의 모델은 E-Class  
자동차의 색상은 blue  
자동차를 주행합니다.  
자동차의 속도는 60  
>>> |
```

```
class Car :  
    def drive(self) :  
        self.speed = 60  
  
myCar = Car() # 객체 생성  
myCar.speed = 0  
myCar.model = "E-Class"  
myCar.color = "blue"  
myCar.year = "2017"  
  
print("자동차 객체를 생성하였습니다.")  
print("자동차의 속도는", myCar.speed)  
print("자동차의 모델은", myCar.model)  
print("자동차의 색상은", myCar.color)  
  
print("자동차를 주행합니다.")  
myCar.drive()  
print("자동차의 속도는", myCar.speed)
```

# 자동차 객체 생성 및 초기화

---

- def \_\_init\_\_()
  - 객체가 생성될 때 객체를 초기화하는 방법
  - 객체의 속성에 따라 파라미터를 설정하여 초기화 진행

```
class Car :  
    def __init__(self, speed, model, color) :  
        self.speed = speed  
        self.model = model  
        self.color = color  
  
    def drive(self) :  
        self.speed = 60  
  
myCar = Car(0, "E-Class", "blue") # 객체 생성과 초기화  
print("자동차 객체를 생성하였습니다.")  
  
print("자동차의 속도는", myCar.speed)  
print("자동차의 모델은", myCar.model)  
print("자동차의 색상은", myCar.color)  
  
print("자동차를 주행합니다.")  
myCar.drive()  
print("자동차의 속도는", myCar.speed)
```



# 하나의 클래스로 여러 개의 객체 생성

---

- 동일한 클래스로 여러 개의 객체 생성 가능
- 객체마다 다른 값의 속성을 설정할 수 있음

```
class Car :  
    def __init__(self, speed, model, color) :  
        self.speed = speed  
        self.model = model  
        self.color = color  
  
    def drive(self) :  
        self.speed = 60  
  
dadCar = Car(0, "A6", "silver")  
momCar = Car(0, "520d", "white")  
myCar = Car(0, "E-Class", "blue")  
  
print(dadCar.color)  
print(momCar.color)  
print(myCar.color)
```

silver  
white  
blue

# 객체 print

- `__str__()`
  - 객체를 `print()`로 출력할 때 자동으로 호출됨
  - 객체의 데이터를 문자열로 만들어서 반환

```
class Car :
    def __init__(self, speed, model, color) :
        self.speed = speed
        self.model = model
        self.color = color

    def __str__(self) :
        msg = "속도 : "+str(self.speed)+" 모델 : "+self.model+" 색상 : "+self.color
        return msg

    def drive(self) :
        self.speed = 60

dadCar = Car(0, "A6", "silver")
momCar = Car(0, "520d", "white")
myCar = Car(0, "E-Class", "blue")

print(myCar)
myCar.drive()
print("자동차의 속도는", myCar.speed)

print(momCar)
momCar.drive()
print("자동차의 속도는", momCar.speed)
```

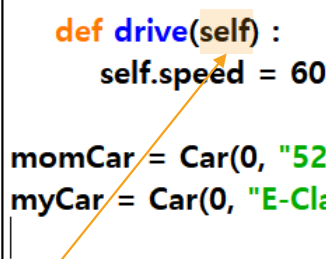
속도 : 0 모델 : E-Class 색상 : blue  
자동차의 속도는 60  
속도 : 0 모델 : 520d 색상 : white  
자동차의 속도는 60

# self란?

---

- self 매개 변수
  - 어떤 객체가 메소드를 호출했는지 알려줌
  - 객체이름.메소드()에서 객체 이름이 **self**로 전달

```
class Car :  
    def __init__(self, speed, model, color) :  
        self.speed = speed  
        self.model = model  
        self.color = color  
  
    def __str__(self) :  
        msg = "속도 : "+str(self.speed)+" 모델 : "+self.model+" 색상 : "+self.color  
        return msg  
  
    def drive(self) :  
        self.speed = 60  
  
momCar = Car(0, "520d", "white")  
myCar = Car(0, "E-Class", "blue")  
|  
momCar.drive()  
myCar.drive()
```



# 은행 계좌 클래스 및 객체 생성

- 속성 : owner, balance
- 메소드 : deposit(), printbalance(), print\_account()

```
class Account :
    def __init__(self, name, balance) :
        self.owner = name
        self.balance = balance

    def deposit(self, amount) :
        self.balance = self.balance + amount

    def printbalance(self) :
        print(self.balance)

    def print_account(self) :
        print("계좌 소유주 : %s\n현재 잔액 : %s" % (self.owner, self.balance))

acc1 = Account("이경민", 50000)
acc1.print_account()
acc1.deposit(1000)
acc1.printbalance()
```

계좌 소유주 : 이경민  
현재 잔액 : 50000  
51000

## LAB10 실습과제 (**bank.py**)

---

- **클래스**를 이용한 은행 계좌 프로그램
- **Account** 클래스를 활용하여 잔액조회, 입금, 출금, 종료가 가능한 ATM 프로그램 작성
- 4개의 **메소드** 작성
  - 1. 초기화 - 계좌 소유자와 초기 금액(0)을 인자로 받음
  - 2. 입금(deposit)
  - 3. 출금(withdraw)
  - 4. 계좌 조회(print\_account)
- 메뉴를 출력하는 함수 생성
- 메뉴 잘못 선택 시 재입력 받기

## LAB10 실습과제 (bank.py)

---

- [실행화면]

계좌 소유자의 이름을 입력해주세요 : |

계좌 소유자의 이름을 입력해주세요 : 이경민

\*\*\*\*\*

**Sookmyung Bank ATM**

\*\*\*\*\*

1. 잔액확인
2. 입금
3. 출금
4. 종료

\*\*\*\*\*

메뉴를 선택해주세요 >>> |

# LAB10 실습과제 (bank.py)

- 메뉴 잘못 선택 시 재입력 받기
- 초기 잔액 확인 시 0원

메뉴를 선택해주세요 >>> 6

메뉴를 다시 선택해주세요

\*\*\*\*\*

## Sookmyung Bank ATM

\*\*\*\*\*

1. 잔액확인
2. 입금
3. 출금
4. 종료

\*\*\*\*\*

메뉴를 선택해주세요 >>> |

\*\*\*\*\*

## Sookmyung Bank ATM

\*\*\*\*\*

1. 잔액확인
2. 입금
3. 출금
4. 종료

\*\*\*\*\*

메뉴를 선택해주세요 >>> 1

계좌 소유자 : 이경민

현재 잔액 : 0

\*\*\*\*\*

## Sookmyung Bank ATM

\*\*\*\*\*

1. 잔액확인
2. 입금
3. 출금
4. 종료

\*\*\*\*\*

메뉴를 선택해주세요 >>> |

# LAB10 실습과제 (bank.py)

- 잘못된 금액을 입력할 경우, 안내 메시지 출력 후 입금 안되어야함
- 입금에 성공했을 경우 안내 메시지 출력

```
*****
Sookmyung Bank ATM
*****
1. 잔액확인
2. 입금
3. 출금
4. 종료
*****

메뉴를 선택해주세요 >>> 2

입금할 금액을 입력해주세요 : -50000
정확한 금액을 입력해주세요.

*****
Sookmyung Bank ATM
*****
1. 잔액확인
2. 입금
3. 출금
4. 종료
*****

메뉴를 선택해주세요 >>> |
```

```
*****
Sookmyung Bank ATM
*****
1. 잔액확인
2. 입금
3. 출금
4. 종료
*****

메뉴를 선택해주세요 >>> 2

입금할 금액을 입력해주세요 : 40000
40000원이 성공적으로 입금되었습니다.

*****
Sookmyung Bank ATM
*****
1. 잔액확인
2. 입금
3. 출금
4. 종료
*****

메뉴를 선택해주세요 >>>
```



# LAB10 실습과제 (bank.py)

- 출금도 입금과 마찬가지로 상황에 따라 안내 메시지 출력
- 잔액보다 큰 금액을 인출할 경우, 안내 메시지 출력 후 출금 안 되어야함

```
*****
Sookmyung Bank ATM
*****
1. 잔액확인
2. 입금
3. 출금
4. 종료
*****

메뉴를 선택해주세요 >>> 3

출금할 금액을 입력해주세요 : -3000
정확한 금액을 입력해주세요.

*****
Sookmyung Bank ATM
*****
1. 잔액확인
2. 입금
3. 출금
4. 종료
*****

메뉴를 선택해주세요 >>> |
```

```
*****
Sookmyung Bank ATM
*****
1. 잔액확인
2. 입금
3. 출금
4. 종료
*****

메뉴를 선택해주세요 >>> 3

출금할 금액을 입력해주세요 : 1000000000000000
잔액 부족. 거래 거절되었습니다.

*****
Sookmyung Bank ATM
*****
1. 잔액확인
2. 입금
3. 출금
4. 종료
*****

메뉴를 선택해주세요 >>> |
```

```
*****
Sookmyung Bank ATM
*****
1. 잔액확인
2. 입금
3. 출금
4. 종료
*****

메뉴를 선택해주세요 >>> 3

출금할 금액을 입력해주세요 : 19000
19000원이 성공적으로 인출되었습니다.
```

# LAB10 실습과제 (bank.py)

- 종료 시 메시지와 함께 프로그램 종료

```
*****
Sookmyung Bank ATM
*****

1. 잔액확인
2. 입금
3. 출금
4. 종료
*****

메뉴를 선택해주세요 >>> 1

계좌 소유자 : 이경민
현재 잔액 : 21000

*****
Sookmyung Bank ATM
*****

1. 잔액확인
2. 입금
3. 출금
4. 종료
*****

메뉴를 선택해주세요 >>> 4

이용해주셔서 감사합니다.
>>>
```

# 과제 채점 기준·기한

---

- 과제 제출 기한
  - 6월 16일 화요일 오후 11시까지 제출
- 제출 장소
  - 스노우보드 해당 주차 과제 제출 페이지에 업로드
- 추가 제출
  - 제출기한 이후 24시간 이내 메일로 전송 : 2점 감점
  - 그 이후는 받지 않음
- 표절X

# 과제 제출 형식 & 질문 메일

---

- **제출물** : 소스파일(.py)과 과제보고서(.docx) 합친 **압축파일 제출**
- **소스파일 이름** : 매 실습 과제 마다 ppt에 제시 예정
- **과제 보고서 양식** : **스노우보드**에서 다운로드
- **(소스파일+과제보고서) 압축 파일 이름** : **Lab10\_학번\_이름**
  
- **이메일** : newkml22@gmail.com
- **질문 시 주의사항**
  - **과목, 분반, 전공, 이름, 학번** 알려주세요.
  - 몇 번 과제에서, 어떤 부분이 막혔는지 어떤 과정인지 **설명**과 함께 보내주세요.
  - 출석 문의, 과제 늦은 제출도 메일로
  - 답장까지 시간이 걸릴 수도 있으니 제출 과제 질문은 미리 해주세요!