# 파이썬과 데이터 엔지니어링

2020년, June

# 변수 (Variable)

- 변수 할당 연산자 =

  ```
  변수명 = 값
  ```

- 동적 타이핑 : 파이썬은 자료형을 명시해 주지 않아도 리터럴 표현에 의해 자료형을 파악한다.

- 객체를 참조한다.

- 메모리에 데이터가 저장된다.

# 자료형(DataType)

| 자료형(Data Type) | 리터럴(literal) | 형변환 내장 함수 |
|---|---|---|
| 숫자(Number) - int | 정수 숫자 (예. 1, 10, -2) | int() |
| 숫자(Number) - float | 실수 숫자 (예. 1.3, 30., 8) | float() |
| Boolean | True, False | bool() |
| 문자열(String) | 따옴표 “ ”, ‘ ’ | str() |
| 리스트(List) | [ ] | list() |
| 튜플(Tuple) | ( ) | tuple() |
| 딕셔너리(Dictionary) | {key:value} | dict() |
| 셋(Set) | { } | set() |

# 자료형(DataType)

- **Data Structure, Container, Collections**
  - list, tuple, dict, set
- **Sequential Type**
  - string, list, tuple, set (range객체…)
- **Iterable**
  - string, list, tuple, dict, set (range객체, dict의 key,value, item, file….)
- **Immutable(불변)**
  - tuple, string

# 자료형(DataType) - string 관련 메소드

| 메소드 | 동작 |
| --- | --- |
| s.upper() | 대문자로 변환 |
| s.lower() | 소문자로 변환 |
| s.split() | 구분자를 기준으로<br>문자열을 나누어 리스트로<br>반환 |
| s.join(list) | 구분자를 기준으로<br>리스트의 요소를 문자열로<br>합친다. |
| s.strip() | 공백 제거 |
| s.find() | 글자의 첫번째 인덱스를<br>반환 |
| s.title() | 첫번째 글자를 대문자로 |

# 자료형(DataType) - list 관련 메소드

| 메소드 | 동작 |
| --- | --- |
| l.append() | 리스트 요소 추가 |
| l.extend() | 리스트 확장 |
| l.insert() | 원하는 위치에 리스트 삽입 |
| l.pop() | 리스트 요소 제거 (index) |
| l.remove() | 리스트 요소 제거 (값) |
| l.sort() | 리스트 정렬 |
| l.reverse() | 리스트 역순으로 정렬 |

# 자료형(DataType) - dict, set 관련 메소드

| 메소드 | 동작 |
| --- | --- |
| d.keys() | dictionary의 key값만 반환 |
| d.values() | dictionary의 value값만 반환 |
| d.items() | dictionary의 item값만 반환 |
| set.add() | set에 요소 추가 |
| set.update() | set에 요소를 갱신한다. |
| set.union() | 두 집합의 합집합을 만든다. |
| set.intersection() | 두 집합의 교집합을 만든다. |
| set.difference() | 두 집합의 차집합을 만든다. |

# [] 대괄호 이용

- **Sequential Type**
  - list, tuple, string, set…
  - Index 값으로 접근
  - [index], [start:end], [start:end:stride]

- **Dictionary**
  - Key 값으로 접근
  - [key값]

# 자주 사용되는 파이썬 내장 함수

- **자료형**
  - 형변환 함수들...
  - type() : 자료의 형(type) 반환
  - len() : 자료의 크기 반환
- sorted( )
- input( ), print( )
- open( )
- range( )
- id()
- format( ) : 문자열 메소드는 아니고 formatted 된 타입으로 반환한다. *기본 format 스펙은 문자열이다.

https://docs.python.org/3/library/functions.html

# 연산자

- 산술 연산자
  - +, -, *, /, //, %, **
- 논리 연산자
  - and(&), or(|)
- 비교 연산자
  - **==, !=, >, <, >=, <=, in**

- 산술 연산자 : +, *(정수곱)
  - string
  - list

# 코드 구조

- 들여쓰기 (indentation)
- 반복문(for문, while문)
- 조건문(if, else, elif)
- break, continue

# 코드 구조 - 반복문(for문)

- Iterable한 객체를 순회한다.

```
for <변수> in <iterable한 객체> :
    <코드>
```

- iterable한 객체, iterate(순회하다)의 의미
  - 파이썬 정수, 실수, Boolean은 iterable하지 않다!

# 코드 구조 - 반복문(while문)

- while문의 조건이 참인 동안 반복한다.

```
while <조건문> :
    <코드>
```

- 조건이 동안 계속 while문이 계속 동작 함으로 무한 loop에 빠지지 않게 주의 하여 설계한다.
    - 무한 loop에 빠지면? kernel을 restart하여 강제로 종료 시킬 수 밖에 없다.

# 코드 구조 - 조건문(if문,elif문,else문)

- if문의 조건이 참이면 if문을 수행하고 if문의 조건이 거짓이면 else를 수행한다.
- elif문을 추가하여 조건을 추가할 수 있다.

```
if <조건1> :
    <코드>
elif <조건2> :
  <코드>
else:
  <코드>
```

# 코드 구조 - continue, break문

- continue: 반복문에서 어떤 조건에서는 코드를 건너뛰고 (skip) 싶을 때 사용 한다.

  `continue`

- break: 반복문에서 어떤 조건에는 코드를 중단(stop)하고 싶을때 사용한다.

  `break`

# 함수

- 정의한다(define)

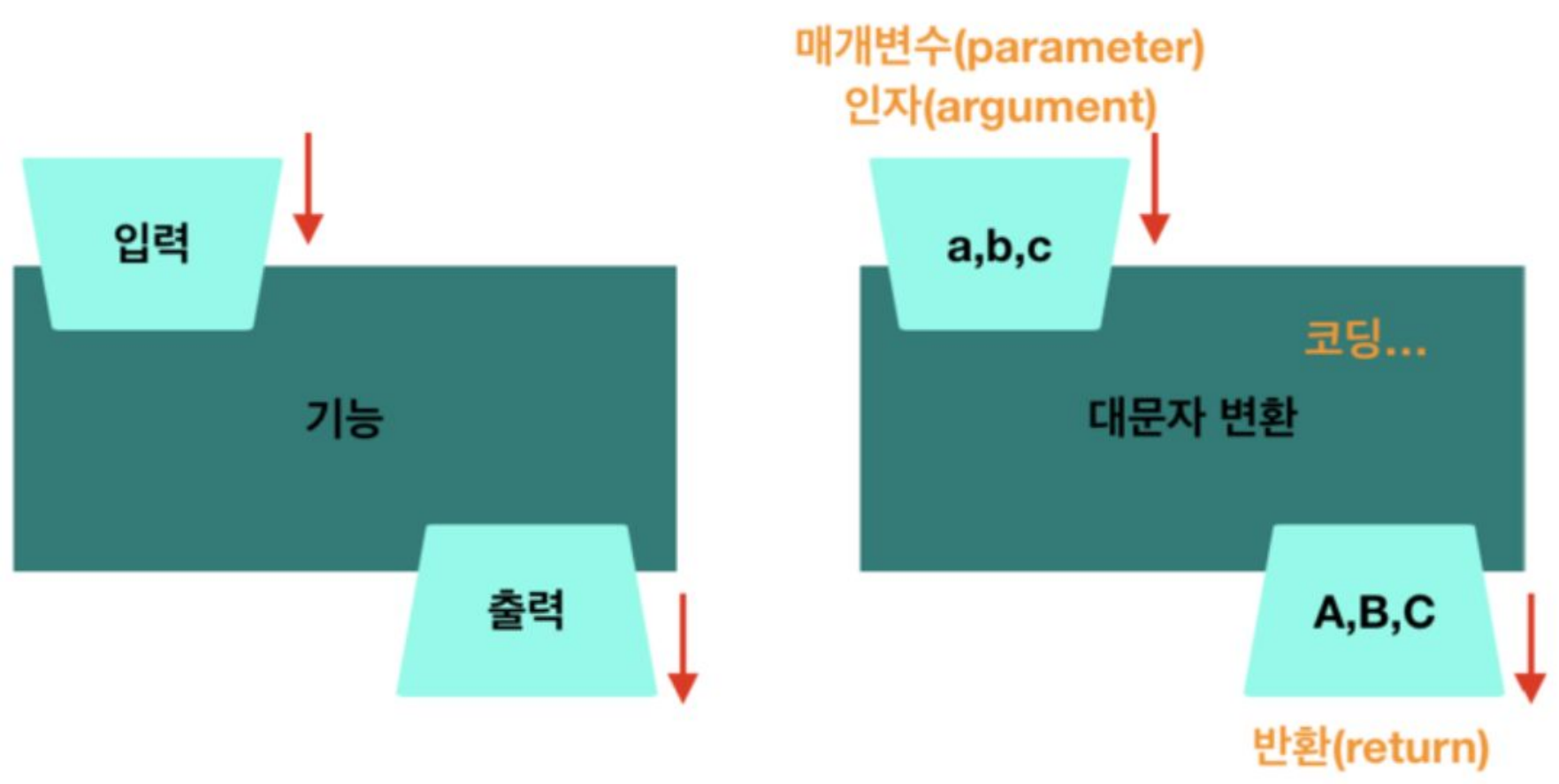
**def 키워드 사용** →

```
def 함수이름():
    <코드>
```

- 호출한다(call)

```
함수이름()
```

# 함수 - 입력과 출력


입력
기능
출력
매개변수(parameter)
인자(argument)
a,b,c
코딩...
대문자 변환
A,B,C
반환(return)

# 함수 - 입력: 매개변수(parameter)와 인자(argument)

- 선언할때 입력값으로 매개변수(parameter)를 설정

```
def 함수이름(a, b):
    print(a+b)
```

→ 매개변수 a,b

- 호출할 때, 입력값으로 인자(argumetn)를 넣는다.

```
함수이름(3, 4)
```

→ 인자 3,4

# 함수 - 출력: 반환값(return)

- 선언할때 함수의 출력값(반환값)으로 return 키워드를 사용하여 설정할 수 있다.

```
def 함수이름(a, b):
    print(a+b)
    return a+b
```

**return 키워드**

- 반환값이 있다는 것?
  함수를 호출한 결과를 대입해 보았을 때, None값이 아닌것을 뜻한다.

# 함수 - 입력과 출력 정리

매개X출력X → def func():  // f = func() 값이 없음.
매개X출력O → return   // f = func() 값이 있음.
매개O출력X → def func(a,b) // f = func(a, b) 값이 없음
매개O출력O → return // f = func(a,b) 값이 있음

# 함수 - 변수의 범위(scope)

- 함수 밖에서 선언된 변수는 함수내에서는 공통으로 사용되지 않는 것을 원칙으로 한다.
  - 함수 안에 선언된 변수를 지역변수(local변수)
  - 함수 밖에서 선언된 변수와 함수 안에서 선언된 변수를 같이 쓰고 싶으면, 전역변수(global변수)로 만들어 준다.

```
c = 30
def 함수이름(a):
    global c
    c += a
    return c
```

함수 밖에 선언된 변수

전역변수 키워드

global 키워드를 적지 않고 c를 선언하면 함수밖에서 선언된 c의 값이 적용되지 않는다.

# 함수사용의 목적

코드의 중복을 피하고
코드를 재사용(reuse)하기
위함.

# 변수와 객체

- 파이썬에서는 모든것이 “객체”로 되어 있다.
- 대부분 객체는 속성과 메소드를 갖는다.

> 원문: Everything in Python is an object, and almost everything has attributes and methods. All functions have a built-in attribute __doc__, which returns the doc string defined in the function's source code.
> https://linux.die.net/diveintopython/html/getting_to_know_python/everything_is_an_object.html

- 이 말의 뜻은 파이썬에서는 모든것(숫자, 문자열, list, tuple,등 자료형, 함수, 모듈등등...)이 다 객체이다.
    - 예) string은 메소드로 join, split등이 있었다.

- 변수는?

# 변수와 객체

- 변수는 객체를 참조할 뿐이다., 변수는 객체의 이름일 뿐이다
- 변수명 a에 숫자 객체 3을 할당한다는 의미는...? 단지 객체에 이름을 붙이는 행위에 지나지 않는다.

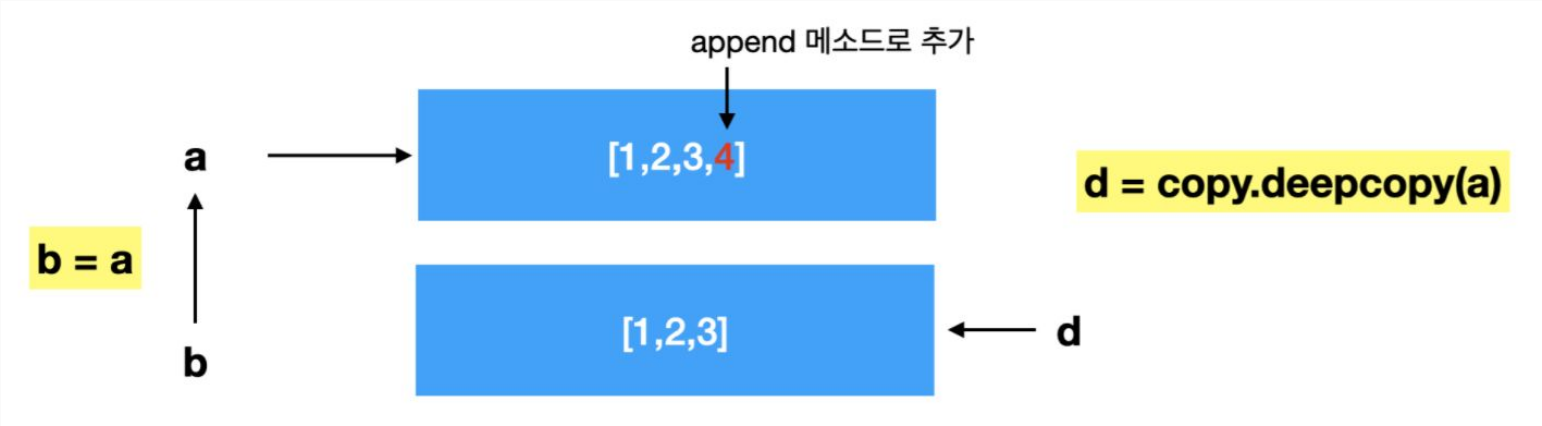
- id()함수로 변수명a와 숫자객체3은 같은 객체임을 알 수 있다.


# 변수 - 얕은 복사

- 변수는 객체를 참조할 뿐이다., 변수는 객체의 이름일 뿐이다
- 변수명 a에 숫자 객체 3을 할당한다는 의미는...? 단지 객체에 이름을 붙이는 행위에 지나지 않는다.

- id()함수로 변수명a와 숫자객체3은 같은 객체임을 알 수 있다.

# 변수 - 깊은 복사




- 실제로 객체를 복사하고 싶을때는 어떻게 해야 할까?
  - [1,2,3] 객체를 변수 a에 할당한다.
  - [1,2,3] 객체를 복사 한다.
  - 복사한 [1,2,3] 객체에 변수 d를 할당한다.
  - 변수 a의 [1,2,3]에 요소 4를 추가한다. (=변수 d와 변수 a는 다르다. )

# 변수 - 얕은 복사, 깊은 복사

- 객체 참조에 의한 복사를 얕은 복사(Shallow Copy)라고 한다.
  - 같은 객체에 변수 할당을 중복한다.
  - copy.copy()메소드를 통해 얕은 복사를 할 수 있다.

- 실제로 객체를 복사하는 것을 깊은 복사(Deep Copy)라고 한다.
  - copy.deepcopy()메소드를 통해 깊은복사를 할 수 있다.

# 클래스

- 파이썬 클래스
  - 사용자 정의 객체를 만들고 싶을때, 클래스를 사용하여 객체의 설계도를 만들 수 있다.

- 클래스에 의해 만들어진 객체를 인스턴스 객체(instance object) 라고 한다.

# 클래스 - 선언과 사용

- 클래스 선언

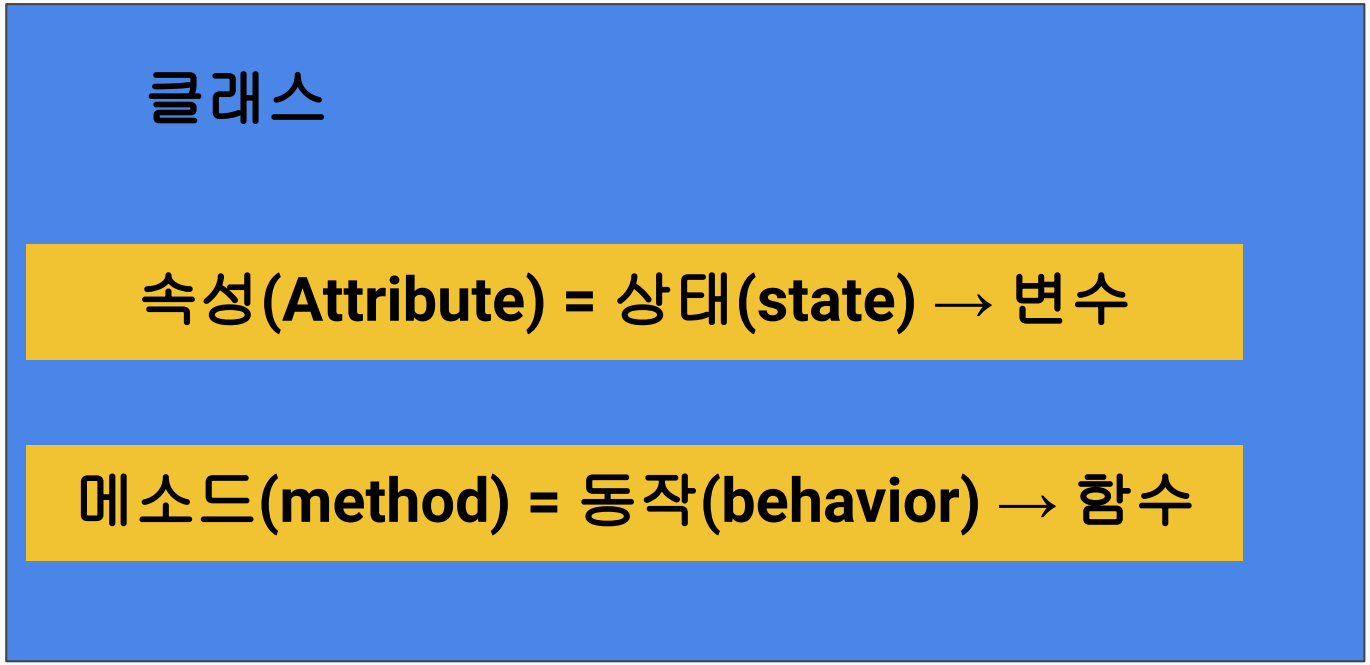
```
class 클래스명:
    <코드>
```

- 클래스 사용 (instantiation):

```
클래스명()
```

# 클래스 - 속성과 메소드

“앞서, 객체는 속성과 메소드를 갖는다고 했다. 클래스로 객체를 정의할 때 속성과 메소드를 정의할 수 있습니다 ”



# 클래스 - 속성과 메소드 코드 예제

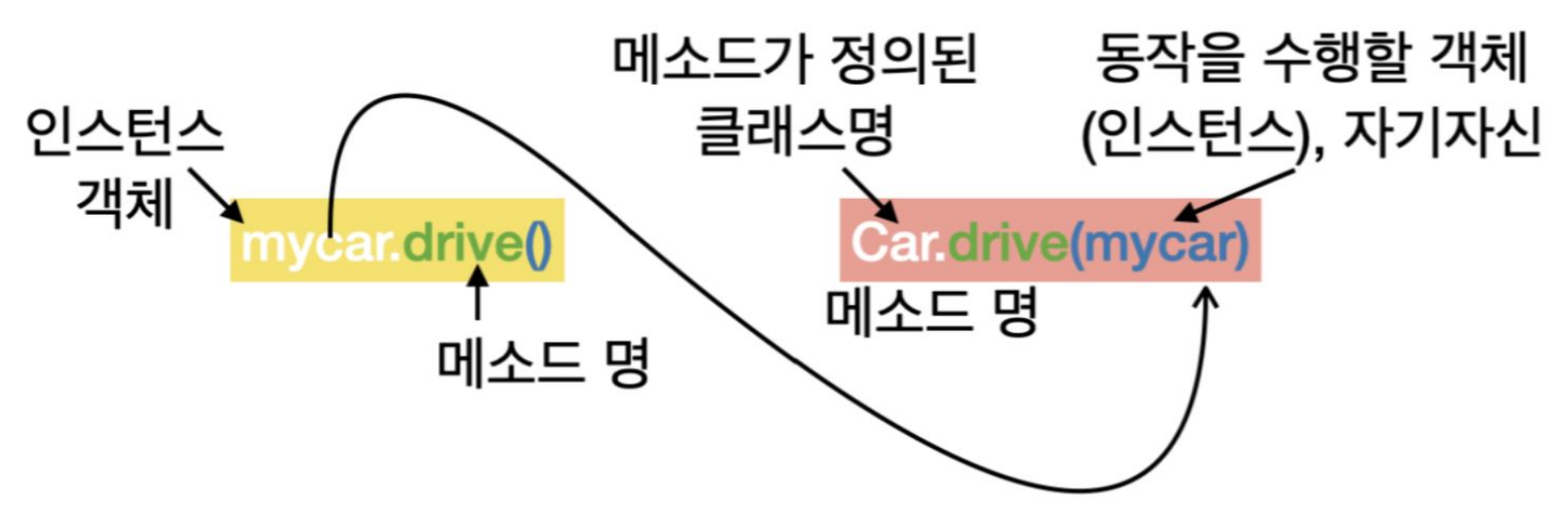
```
class car:
    color = ‘red’
    category = ‘sedan’
```

```
class car:
    def drive(self):
        print(“driving”)
    def accel(self, speed_up, current_speed=10):
        self.current_speed = current_speed
        self.speed_up = speed_up
        self.current_speed += self.speed_up
        print("speed up {} driving at{}".format(self.speed_up,  self.current_speed))
```

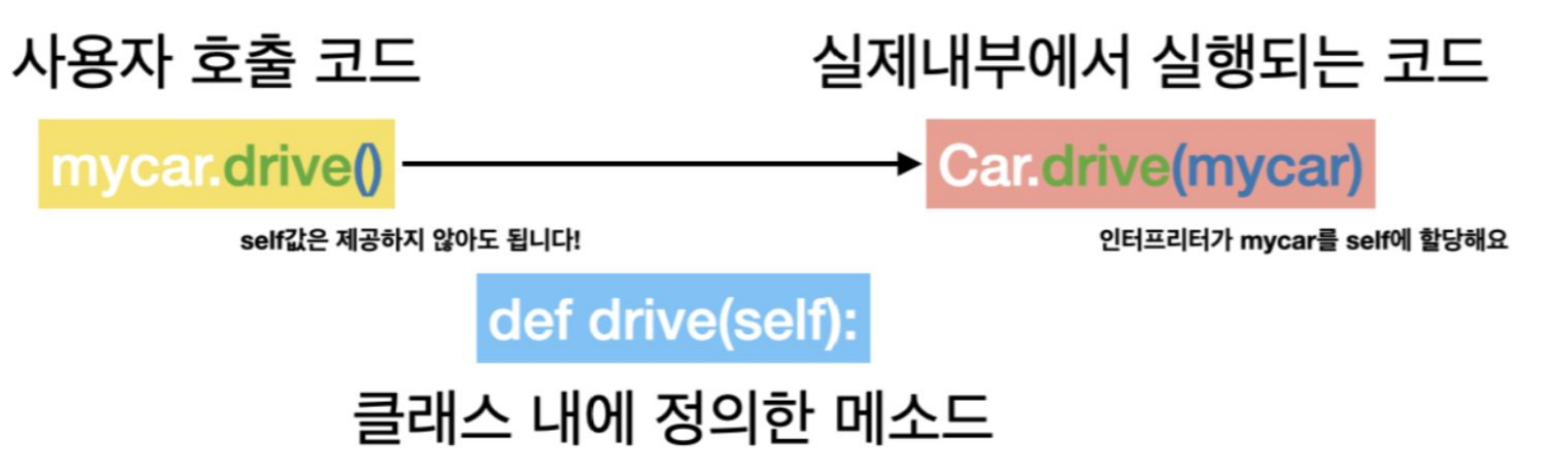
# 클래스 - self

- self, 자기 자신,
- 클래스의 메소드에서 첫번째 매개변수(parameter)는 self를 넣어준다.
- 메소드를 호출할 때 인자(argument)에는 대입하지 않아도 된다.
  - 파이썬 인터프리터에 의해 호출된다.
- 메소드 내 정의된 변수는 접두사 self.을 붙여준다.

# 클래스 - self


인스턴스
객체
mycar.drive()
메소드 명
메소드가 정의된
클래스명
Car.drive(mycar)
메소드 명
동작을 수행할 객체
(인스턴스), 자기자신

# 클래스 - self


사용자 호출 코드
실제내부에서 실행되는 코드
mycar.drive()
Car.drive(mycar)
self값은 제공하지 않아도 됩니다!
인터프리터가 mycar를 self에 할당해요
def drive(self):
클래스 내에 정의한 메소드

# 클래스 - 생성자(constructor) __init__

- __init__ 메소드안에 인자를 전달함로써 인스턴스 객체의 속성을 초기화 할 수 있다 .

  ```
  def __init__(self):
  ```

- 이를 인스턴스 객체의 초기화(initialized instance) 라고 히고, __init__함수는 생성자(constructor)라고 한다.

**<Comment>**

- __init__ 역시 def 키워드로 정의합니다.
- 다른 객체 지향 언어에서는 생성자는 객체 인스턴스화와 초기화 2가지 작업 수행합니다.
- 그러나 파이썬의 생성자는 초기화만 수행 합니다. 객체 인스턴스화는 앞서 말한 클래스 사용시 변수 할당을 통해 이루어집니다.
- 그리고 이 __init__처럼 앞뒤에 언더바(_)가 두개씩 있는 메소드를 매직 메소드라고 합니다.

# 클래스 - 클래스변수, 인스턴스변수

**클래스 변수**

- 클래스에 선언된 속성을 클래스 변수라고 하며 클래스에 의해 생성된 **모든 객체에서 같은 값을 조회**할 때 가능하다.
- (Public)

**인스턴스변수**

- __init__() 안에 선언된 변수를 인스턴스 변수라고 합니다. 객체가 인스턴스화 될 때마다 새로운 값이 할당되며 서로 다른 객체 간에는 값을 공유할 수 없다.
- (Private)

객체 단위로 변경되는 변수는 인스턴스 변수로 선언합니다!

# Class 상속

- 상속 방법

```
class A:
  pass
class B(A):
  pass
```

사전에 정의된 클래스 A를 B에 상속했다.

- **메소드 추가** : 부모클래스에 없는 메소드를 정의
- **메소드 오버라이드** : 부모클래스에 있는 메소드를 재 정의
- **super()** : 부모클래스에 있는 메소드를 호출

# 객체를 사용하는 이유

## 추상화(abstraction)

- 컴퓨터 과학에서 추상화(abstraction)는 복잡한 자료, 모듈, 시스템 등으로부터 핵심적인 개념 또는 기능을 간추려 내는 것을 말한다.

## 캡슐화(영어: encapsulation)

- 객체의 속성(data fields)과 행위(메서드, methods)를 하나로 묶는다.
- 실제 구현 내용 일부를 외부에 감추어 은닉한다. (-> 모듈, 패키지, 라이브러리)

https://github.com/python/cpython/blob/3.8/Lib/collections/__init__.py

# 객체를 사용하는 이유

**Account**

속성
- 잔액

메소드
- 잔액조회
- 입금
- 출금

account1 = Account()

account2 = Account()

<comment>

복잡한 기능은 미리 구현해 놓고 클래스로 정의하면 필요할 때 인스턴스 객체로 만들어 사용한다.

## NumPy 특징

- 강력한 N 차원 배열 객체
- 정교한 브로드케스팅(Broadcast) 기능
- C/C ++ 및 포트란 코드 통합 도구
- 유용한 선형 대수학, 푸리에 변환 및 난수 기능
- 범용적 데이터 처리에 사용 가능한 다차원 컨테이너

| Numpy 자료형(Data Type) |
| --- |
| ndarray() 객체 |

# 배열? list사용한다고 하면...?

**배열,, list를 행렬 연산을 사용한다고 하면...?**

- 행렬 더하기 list로 표현한다고 하면 어떨까?
- 행렬 상수 곱하기(스칼라곱)을 list로 표현한다면 어떨까?

| NumPy 관련 메소드 | |
|---|---|
| ndarray 만들기 | np.arange()<br>np.array([]) |
| 크기 | np.size<br>np.shape<br>np.dim |
| type | arr.dtype()<br>type(a) |
| 특수행렬 :<br>0 행렬, 1행렬, 단위행렬 | np.zeros([])<br>np.ones |
| 랜덤 | np.random.randint()<br>np.random.choice()<br>np.random.permutaion() |
| 크기 변형 | np.reshape()<br>np.flatten() |
| 주요 연산 | 행렬 연산과 브로드 캐스팅<br>np.dot(), np.transpose(), np.sum(), np.sqrt() |

# Pandas - Pandas 라이브러리 없이 CSV파일을 읽는 코드 예시

```
with open(input_file, 'r', newline='') as filereader:
    with open(output_file, 'w', newline='') as filewriter:
        header = filereader.readline()
        header = header.strip()
        header_list = header.split(',')
        print(header_list)
        filewriter.write(','.join(map(str,header_list))+'\n')
        for row in filereader:
            row = row.strip()
            row_list = row.split(',')
            print(row_list)
            filewriter.write(','.join(map(str,row_list))+'\n')
```

# Pandas

- 통계 분석을 위한 ***R***의 ***DataFrame*** 데이터 타입과 같은 ***Pandas DataFrame***을 사용
- Pandas DataFrame
  - ***테이블 형식의 데이터*** (tabular, rectangular grid 등으로 불림)를 다룰 때 사용
  - Column, Row(데이터), Index
  - Numpy의 ndarray, Pandas의 DataFrame, Series, Python의 dictionary, list 등으로 부터 생성 가능

```
df = pd.DataFrame(np.array([[1, 2, 3], [4, 5, 6]]))
print(df.shape)
print(len(df.index))
print(list(df.columns))
```

```
(2, 3)
2
[0, 1, 2]
```

# Pandas에서 열을 선택하는 방법

- Dataframe에서 특정 컬럼이나 로우 선택
  - iloc, loc

```
 9  df = pd.DataFrame({"A": [1,4,7], "B": [2,5,8], "C":[3,6,9]})
10  print(df.loc[0])
11
12  print(df.loc[:, 'A'])
13  print(df.loc[1:, 'B'])
```

```
A    1
B    2
C    3
Name: 0, dtype: int64
```

```
0    1
1    4
2    7
Name: A, dtype: int64
```

```
1    5
2    8
Name: B, dtype: int64
```

# pandas에서 csv파일 읽기

```
data_frame = pd.read_csv(input_file)
print(data_frame)
data_frame.to_csv(output_file, index=False)
```

| | Supplier Name | Invoice Number | Part Number | Cost | Purchase Date |
|---|---|---|---|---|---|
| 0 | Supplier X | 001-1001 | 2341 | $500.00 | 1/20/14 |
| 1 | Supplier X | 001-1001 | 2341 | $500.00 | 1/20/14 |
| 2 | Supplier X | 001-1001 | 5467 | $750.00 | 1/20/14 |
| 3 | Supplier X | 001-1001 | 5467 | $750.00 | 1/20/14 |
| 4 | Supplier Y | 50-9501 | 7009 | $250.00 | 1/30/14 |
| 5 | Supplier Y | 50-9501 | 7009 | $250.00 | 1/30/14 |
| 6 | Supplier Y | 50-9505 | 6650 | $125.00 | 2/3/14 |
| 7 | Supplier Y | 50-9505 | 6650 | $125.00 | 2/3/14 |
| 8 | Supplier Z | 920-4803 | 3321 | $615.00 | 2/3/14 |
| 9 | Supplier Z | 920-4804 | 3321 | $615.00 | 2/10/14 |
| 10 | Supplier Z | 920-4805 | 3321 | $615.00 | 2/17/14 |
| 11 | Supplier Z | 920-4806 | 3321 | $615.00 | 2/24/14 |

# pandas 필터링 예시

- 특정 조건을 충족하는 행을 필터링
- loc()
  - 특정 행과 열을 동시에 선택

```
data_frame.loc[(data_frame['Supplier Name'].str.contains('Z')) | (data_frame['Cost'] > 600.0), :]
```

```
Supplier Name,Invoice Number,Part Number,Cost,Purchase Date
Supplier X,001-1001,5467,$750.00 ,1/20/14
Supplier X,001-1001,5467,$750.00 ,1/20/14
Supplier Z,920-4803,3321,$615.00 ,2002-03-14
Supplier Z,920-4804,3321,$615.00 ,2002-10-14
Supplier Z,920-4805,3321,$615.00 ,2/17/14
Supplier Z,920-4806,3321,$615.00 ,2/24/14
```

# 파이썬 (csv파일)에서 필터링 예시

- 특정 조건을 충족하는 행을 필터링

  ex) 비용이 특정 값을 초과하는 모든 행을 선택하여 데이터셋으로 만들 경우

  ex) 구매 일자가 특정 날짜 이전인 모든 행을 데이터셋으로 만들 경우

- Supplier Name이 **Supplier Z** or **Cost가 $600.00 이상**인 행만 필터링 → 파일 출력

```
for row_list in filereader:
    supplier = str(row_list[0]).strip()
    cost = str(row_list[3]).strip('$').replace(',', '')
    if supplier == 'Supplier Z' or float(cost) > 600.0:
        filewriter.writerow(row_list)
```

```
Supplier Name,Invoice Number,Part Number,Cost,Purchase Date
Supplier X,001-1001,5467,$750.00 ,1/20/14
Supplier X,001-1001,5467,$750.00 ,1/20/14
Supplier Z,920-4803,3321,$615.00 ,2002-03-14
Supplier Z,920-4804,3321,$615.00 ,2002-10-14
Supplier Z,920-4805,3321,"$6,015.00",2/17/14
Supplier Z,920-4806,3321,"$1,006,015.00 ",2/24/14
```