1. 파이썬의 타입

- 파이썬이 처리할 수 있는 데이터의 종류
- 내장 타입
  - 값 1개 : int, float, str, bool
  - 값 여러개를 나열 : list, tuple, set
  - 이름과 값(키와 값)의 쌍 : dictionary
- 사용자 정의 타입
  - 클래스라는 설계도를 이용해 작성할 수 있다
  - 클래스는 데이터와 데이터를 처리하는 메서드를 하나로 묶은 설계도

2. 클래스

- 파이썬에서 클래스는 타입의 한 종류
- 사용자가 정의한 클래스는 새로운 타입을 생성하는 것과 같다

```
class Circle:
  # 클래스의 변수(객체) 생성을 담당하는 메서드(생성자)
  def __init__(self, radius):
    self.radius = radius

  # 원의 면적 계산: πr^2
  def get_area(self):
    return 3.14 * self.radius * self.radius

  # 원의 둘레 계산: 2πr
  def get_circumference(self):
    return 2 * 3.14159 * self.radius
```

```
# 원 객체 생성
circle = Circle(5)

# 객체 타입 출력
print(type(circle))          # class Circle

# 원의 면적 출력
print(circle.get_area())  # 원의 면적 계산 및 출력
```

3. 게시판의 경우

- dictionary는 값을 키와 쌍으로 구성하는 객체 타입(개발자는 설계도를 직접 만들지 않는다)
  - 클래스에 비해 빠르게 코드를 작성할 수 있다

```
board = dict(bno=1, title='1번글', content='냉무', nickname='spring', readcnt=0)
board['readcnt'] += 1
print(board['readcnt'])
```

- class는 값과 그 값를 다루는 메소드를 설계한 다음 부품화하는 개념
  딕셔너리보다 오류가 적고 재사용이 손쉽다

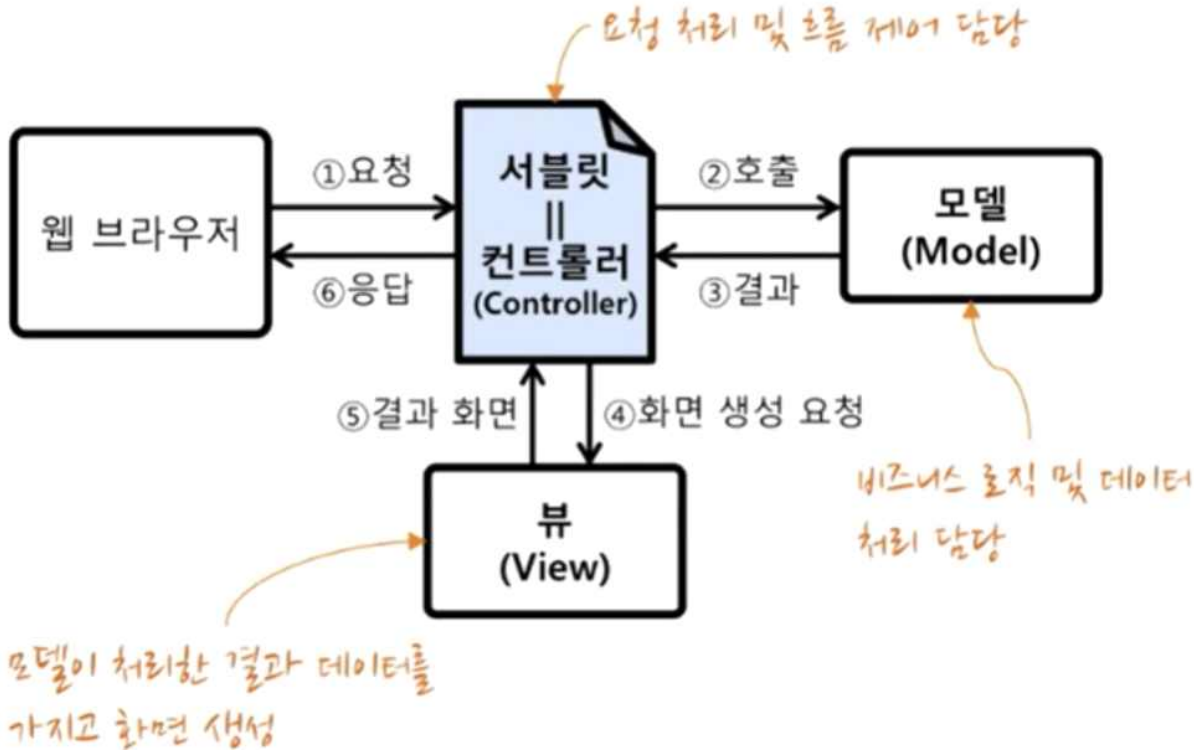
```
class Board:
    def __init__(self, bno, title, content, nickname, readcnt):
        self.bno = bno
        self.title = title
        self.content = content
        self.nickname = nickname
        self.readcnt = readcnt

    def 조회수증가(self):
        self.readcnt += 1
```

```
# 객체 생성
board2 = Board(bno=1, title='1번글', content='냉무', nickname='spring', readcnt=0)
# 메소드 호출
board2.조회수증가()
# 조회수가 1증가 했다
print(board2.readcnt)
```

4. MVC 패턴

- 디자인 패턴 : 개발 중에 발생하던 문제점들에 대한 대응책을 정해놓은 것
- 애플리케이션을 데이터를 처리하는 모델(Model), 사용자에게 보여지는 사용자 인터페이스(UI)를 담당하는 뷰(View), 모델과 뷰를 연결하는 컨트롤러(Controller)로 분리하는 설계 방식
- 노르웨이의 컴퓨터과학자 트리베 린스카우그(Trygve Reenskaug)가 최초로 정의하여, PC부터 웹 애플리케이션까지 광범위하게 사용되는 디자인 패턴이다



- Model
  모든 데이터를 가지고 있어야 한다
  뷰나 컨트롤러에 대해서 전혀 몰라야한다
  데이터 처리를 담당한다
- View
  데이터를 따로 저장해서는 안된다
  모델과 컨트롤러에 대해서 전혀 몰라야한다
  데이터 출력을 담당한다
- Controller
  모델과 뷰에 대해서 알고 있고 연결해 준다
  웹 프로그래밍의 경우 사용자에게 주소를 제공한다
  작업은 항상 컨트롤러에서 시작해서 컨트롤러에서 끝난다