클래스 생성자

Python을 활용한 자료구조 이해하기

붕어빵을 굽는 방식

- 붕어빵 틀로부터 먼저 붕어빵을 굽는다.
- 구어진 붕어빵에 앙꼬를 넣는다. (메서드 호출)

```
class 붕어빵틀:
    def 앙꼬넣기(self, 앙꼬):
        self.앙꼬 = 앙꼬

내빵 = 붕어빵틀()

내빵.앙꼬넣기("딸기맛")
```

생성자

- 클래스로부터 객체가 생성될 때 파이썬 인터프리터에 의해 자동으로 호출되는 특별한 메서드
  - 메모리에 생성된 객체 공간에 데이터를 넣거나 초기화하는 목적으로 사용됨
  - __init__ 이라는 특별한 이름을 가짐
  - 첫 번째 인자로 self를 사용함

```
class 붕어빵틀:
    def __init__(self):     # 어떤 객체를 초기화 할 것인가?
        print("붕어빵 잘 구어짐")
```

객체 생성하기

- 객체를 생성하면 자동으로 생성자가 호출됨
  - 사용자가 호출 하는 것이 아님
  - 파이썬 인터프리터가 객체를 생성한 후 호출하는 것임

```
class 붕어빵틀:
    def __init__(self):
        print("붕어빵 잘 구어짐")

내빵 = 붕어빵틀()
```

붕어빵틀 클래스 업데이트 (1/2)

- 기존에 앙꼬넣기의 역할은 붕어빵이 구어질 때 앙꼬를 넣는 것임
  - 사용자가 객체를 생성한 후 명시적으로 호출했었음 → 생성자로 이름을 변경하면 알아서 호출됨
  - 객체가 생성되면 자동으로 생성자가 호출되는데 생성자는 앙꼬라는 인자가 필요함. 따라서 객체를 생성할 때 구울 붕어빵에 넣을 앙꼬를 같이 전달해야함

```
class 붕어빵틀:
    def 앙꼬넣기(self, 앙꼬):
        self.앙꼬 = 앙꼬

내빵 = 붕어빵틀()
내빵.앙꼬넣기("딸기맛")
```

```
class 붕어빵틀:
    def __init__(self, 앙꼬):
        self.앙꼬 = 앙꼬

내빵 = 붕어빵틀()
```

TypeError: __init__() missing 1 required positional argument: '앙꼬'

붕어빵틀 클래스 업데이트 (2/2)

- 기존에 앙꼬넣기의 역할은 붕어빵이 구어질 때 앙꼬를 넣는 것임
  - 사용자가 객체를 생성한 후 명시적으로 호출했었음 → 생성자로 이름을 변경하면 알아서 호출됨
  - 객체가 생성되면 자동으로 생성자가 호출되는데 생성자는 앙꼬라는 인자가 필요함. 따라서 객체를 생성할 때 구울 붕어빵에 넣을 앙꼬를 같이 전달해야함

```
class 붕어빵틀:
    def 앙꼬넣기(self, 앙꼬):
        self.앙꼬 = 앙꼬

내빵 = 붕어빵틀()
내빵.앙꼬넣기("딸기맛")
```

```
class 붕어빵틀:
    def __init__(self, 앙꼬):
        self.앙꼬 = 앙꼬

내빵 = 붕어빵틀("딸기맛")
print(내빵.앙꼬)
```

연습문제

1. 사람 클래스를 정의하세요.
    - 생성자를 통해 (이름, 생년월일, 성별) 입력
    - 정보출력 메서드에서 정보를 출력

```
>> 첫째 = 사람("유종훈", "19860302", "남")
>> 첫째.정보출력()
1986년 3월 2일 출생
(남) 유종훈
```

연습문제

- 2. 다음과 같이 사용할 수 있는 비행기 클래스를 정의해보세요.

```
>> 비행기_1 = 비행기("보잉787")
>> 비행기_1.이륙()
보잉787 이륙합니다
```

```
>> 비행기_2 = 비행기("에어버스A330")
>> 비행기_2.이륙()
에어버스A330 이륙합니다
```

연습문제

3. 다음과 같이 사용할 수 있는 계좌 클래스를 정의해보세요.
    - 생성자에서 이름과 잔고를 입력
    - 출력 메서드에서 이름과 잔고를 출력

기타 특수 메서드

- underline ('__')은 파이썬에서 사용하는 특수기능

연산자	메서드	설명
+	__add__	덧셈
*	__mul__	곱셈
-	__sub__	뺄셈
/	__truediv__	나눗셈
%	__mod__	나머지
<	__lt__	크다
==	__eq__	같다
len	__len__	길이
str	__str__	문자열 변환

파이썬의 모든 것은 클래스

```
b = str("python")
print(type(b))
print(b.upper())
print(b.count('p'))
```

```
<class 'str'>
PYTHON
1
```