UNIT 34

클래스 사용하기

34 클래스 사용하기

» 클래스 사용하기

- 클래스는 객체를 표현하기 위한 문법임

▼ 그림 게임 캐릭터

34 클래스 사용하기

클래스 사용하기

- 물론 집, 자동차, 나무 등도 클래스로 표현할 수 있음
- 프로그래밍에서는 현실 세계에 있는 개념들뿐만 아니라 컴퓨터 안에서만 쓰이는 개념들도 클래스로 만들어서 표현함

▼ 그림 컴퓨터 안에서만 쓰이는 스크롤 바, 버튼, 체크박스

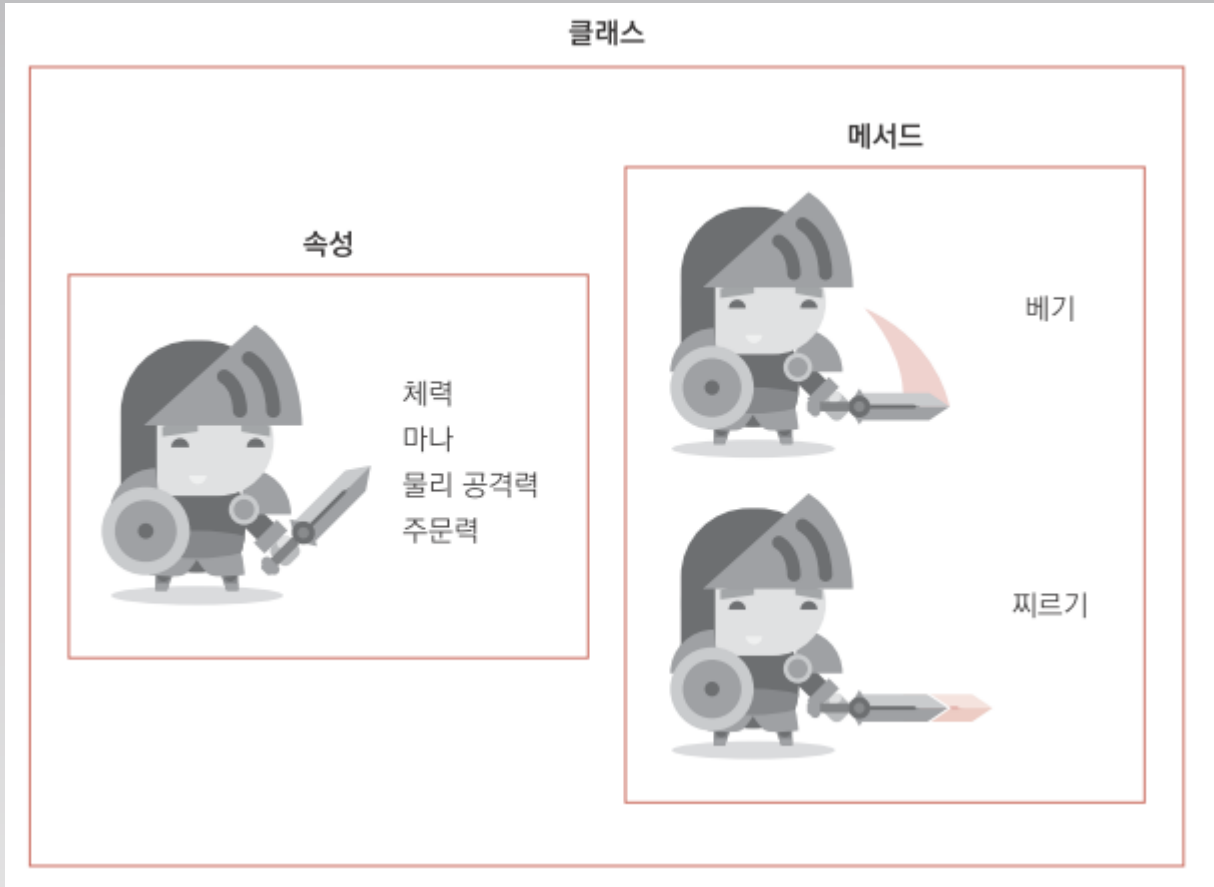
34 클래스 사용하기

» 클래스 사용하기

- 지금까지 나온 기사, 마법사, 궁수, 사제, 집, 자동차, 나무, 스크롤 바, 버튼, 체크 박스처럼 특정한 개념이나 모양으로 존재하는 것을 객체(object)라고 부름
- 프로그래밍으로 객체를 만들 때 사용하는 것이 클래스임
- 체력, 마나, 물리 공격력, 주문력 등의 데이터를 클래스의 속성(attribute)이라 부르고, 베기, 찌르기 등의 기능을 메서드(method)라고 부름

34 클래스 사용하기

▼ 그림 클래스의 속성과 메서드



34 클래스 사용하기

클래스 사용하기

- 프로그래밍 방법을 객체지향(object oriented) 프로그래밍이라고 함
- 객체지향 프로그래밍은 복잡한 문제를 잘게 나누어 객체로 만들고, 객체를 조합해서 문제를 해결함
- 현실 세계의 복잡한 문제를 처리하는데 유용하며 기능을 개선하고 발전시킬 때도 해당 클래스만 수정하면 되므로 유지 보수에도 효율적임

34.1 클래스와 메서드 만들기

» 클래스와 메서드 만들기

- 클래스는 class에 클래스 이름을 지정하고 :(콜론)을 붙인 뒤 다음 줄부터 def로 메서드를 작성하면 됨
- 메서드는 클래스 안에 들어있는 함수를 뜻함
- 파이썬에서는 클래스의 이름은 대문자로 시작하고 메서드 작성 방법은 함수와 같으며 코드는 반드시 들여쓰기를 해야 함(들여쓰기 규칙은 if, for, while과 같음)
- 메서드의 첫 번째 매개변수는 반드시 self를 지정해야 함

```
class 클래스이름:
    def 메서드(self):
        코드
```

- 이제 간단한 사람 클래스를 작성해보자

```
>>> class Person:
...     def greeting(self):
...         print('Hello')
...
```

34.1 클래스와 메서드 만들기

클래스와 메서드 만들기

- 다음과 같이 클래스에 ()(괄호)를 붙인 뒤 변수에 할당함

```
• 인스턴스 = 클래스()
```

```
>>> james = Person()
```

- Person으로 변수 james를 만들었는데 이 james가 Person의 인스턴스(instance)임
- 클래스는 특정 개념을 표현만 할뿐 사용을 하려면 인스턴스를 생성해야 함

34.1 클래스와 메서드 만들기

메서드 호출하기

- 메서드는 클래스가 아니라 인스턴스를 통해 호출함

• 인스턴스.메서드()

```
>>> james.greeting()
Hello
```

- 인스턴스를 통해 호출하는 메서드를 인스턴스 메서드라고 부름

34.1 클래스와 메서드 만들기

파이썬에서 흔히 볼 수 있는 클래스

- 지금까지 사용한 int, list, dict 등도 사실 클래스임
- 우리는 이 클래스로 인스턴스를 만들고 메서드를 사용함

```
>>> a = int(10)
>>> a
10
>>> b = list(range(10))
>>> b
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> c = dict(x=10, y=20)
>>> c
{'x': 10, 'y': 20}
```

34.1 클래스와 메서드 만들기

파이썬에서 흔히 볼 수 있는 클래스

- 인스턴스 b에서 메서드 append를 호출해서 값을 추가함
- 이 부분도 지금까지 메서드를 만들고 사용한 것과 같은 방식임

```
>>> b = list(range(10))
>>> b.append(20)
>>> b
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 20]
```

34.1 클래스와 메서드 만들기

파이썬에서 흔히 볼 수 있는 클래스

- 파이썬에서는 자료형도 클래스임

type(객체)

```
>>> a = 10
>>> type(a)
<class 'int'>
>>> b = [0, 1, 2]
>>> type(b)
<class 'list'>
>>> c = {'x':10, 'y':20}
>>> type(c)
<class 'dict'>
>>> maria = Person()
>>> type(maria)
<class '__main__.Person'>
```

34.1 클래스와 메서드 만들기

인스턴스와 객체의 차이점?

- 보통 객체만 지칭할 때는 그냥 객체(object)라고 부름
- 클래스와 연관지어서 말할 때는 인스턴스(instance)라고 부름
- 다음과 같이 리스트 변수 a, b가 있으면 a, b는 객체임
- a와 b는 list 클래스의 인스턴스임

```
>>> a = list(range(10))
>>> b = list(range(20))
```

34.2 속성 사용하기

속성 사용하기

- 속성(attribute)을 만들 때는 __init__ 메서드 안에서 self.속성에 값을 할당함

```
class 클래스이름:
    def __init__(self):
        self.속성 = 값
```

class_attribute.py

```
class Person:
    def __init__(self):
        self.hello = '안녕하세요.'

    def greeting(self):
        print(self.hello)

james = Person()
james.greeting()    # 안녕하세요.
```

실행 결과

```
안녕하세요.
```

34.2 속성 사용하기

속성 사용하기

```
class Person:
    def __init__(self):
        self.hello = '안녕하세요.'
```

- __init__ 메서드는 james = Person()처럼 클래스에 ( )(괄호)를 붙여서 인스턴스를 만들 때 호출되는 특별한 메서드임
- __init__(initialize)이라는 이름 그대로 인스턴스(객체)를 초기화함
- 앞 뒤로 __(밑줄 두 개)가 붙은 메서드는 파이썬이 자동으로 호출해주는 메서드인데 스페셜 메서드(special method) 또는 매직 메서드(magic method)라고 부름
- 파이썬의 여러 가지 기능을 사용할 때 이 스페셜 메서드를 채우는 식으로 사용하게 됨

34.2 속성 사용하기

속성 사용하기

```
    def greeting(self):
        print(self.hello)
```

```
james = Person()
james.greeting()    # 안녕하세요.
```

- 속성은 __init__ 메서드에서 만든다는 점과 self에 .(점)을 붙인 뒤 값을 할당한다는 점이 중요함
- 클래스 안에서 속성을 사용할 때도 self.hello처럼 self에 점을 붙여서 사용하면 됨

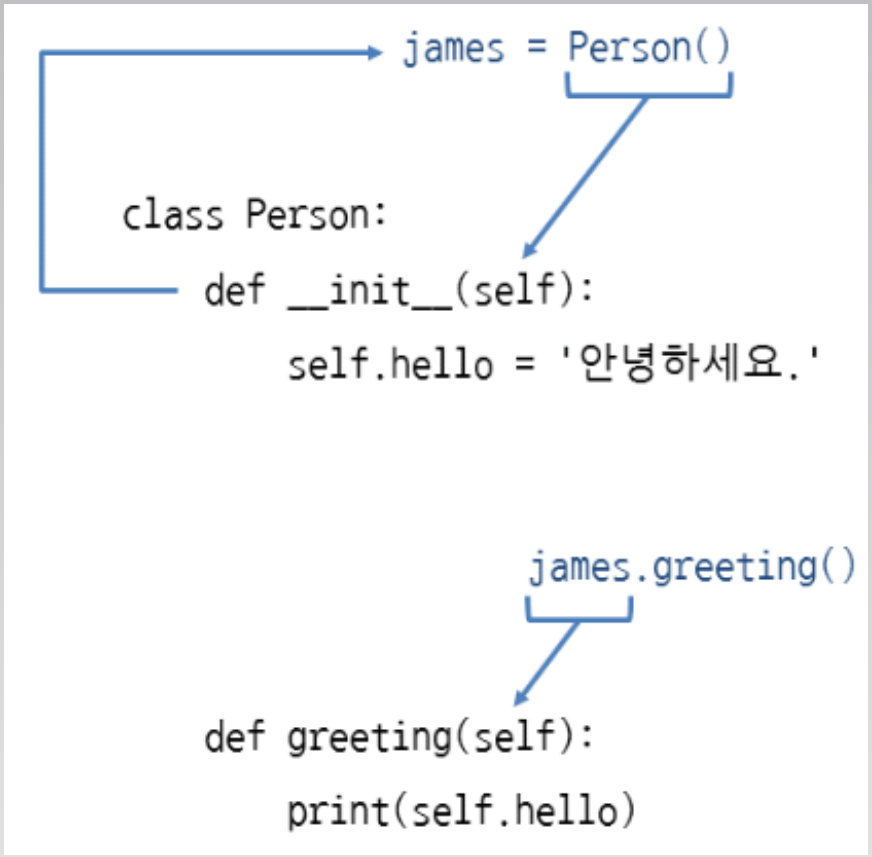
34.2 속성 사용하기

self의 의미

- self는 인스턴스 자기 자신을 의미함
- 우리는 인스턴스가 생성될 때 self.hello = '안녕하세요.'처럼 자기 자신에 속성을 추가함
- __init__의 매개변수 self에 들어가는 값은 Person()이라 할 수 있음
- self가 완성된 뒤 james에 할당됨
- 호출하면 현재 인스턴스가 자동으로 매개변수 self에 들어옴
- greeting 메서드에서 print(self.hello)처럼 속성을 출력할 수 있었던 것임

34.2 속성 사용하기

▼ 그림 인스턴스와 self



34.2 속성 사용하기

인스턴스를 만들 때 값 받기

- 다음과 같이 __init__ 메서드에서 self 다음에 값을 받을 매개변수를 지정함
- 매개변수를 self.속성에 넣어줌

```
class 클래스이름:
    def __init__(self, 매개변수1, 매개변수2):
        self.속성1 = 매개변수1
        self.속성2 = 매개변수2
```

34.2 속성 사용하기

인스턴스를 만들 때 값 받기

- 그럼 Person 클래스로 인스턴스를 만들 때 이름, 나이, 주소를 받아보지

class_init_attribute.py

```
class Person:
    def __init__(self, name, age, address):
        self.hello = '안녕하세요.'
        self.name = name
        self.age = age
        self.address = address

    def greeting(self):
        print('{0} 저는 {1}입니다.'.format(self.hello, self.name))

maria = Person('마리아', 20, '서울시 서초구 반포동')
maria.greeting()    # 안녕하세요. 저는 마리아입니다.

print('이름:', maria.name)       # 마리아
print('나이:', maria.age)        # 20
print('주소:', maria.address)    # 서울시 서초구 반포동
```

실행 결과

```
안녕하세요. 저는 마리아입니다.
이름: 마리아
나이: 20
주소: 서울시 서초구 반포동
```

34.2 속성 사용하기

인스턴스를 만들 때 값 받기

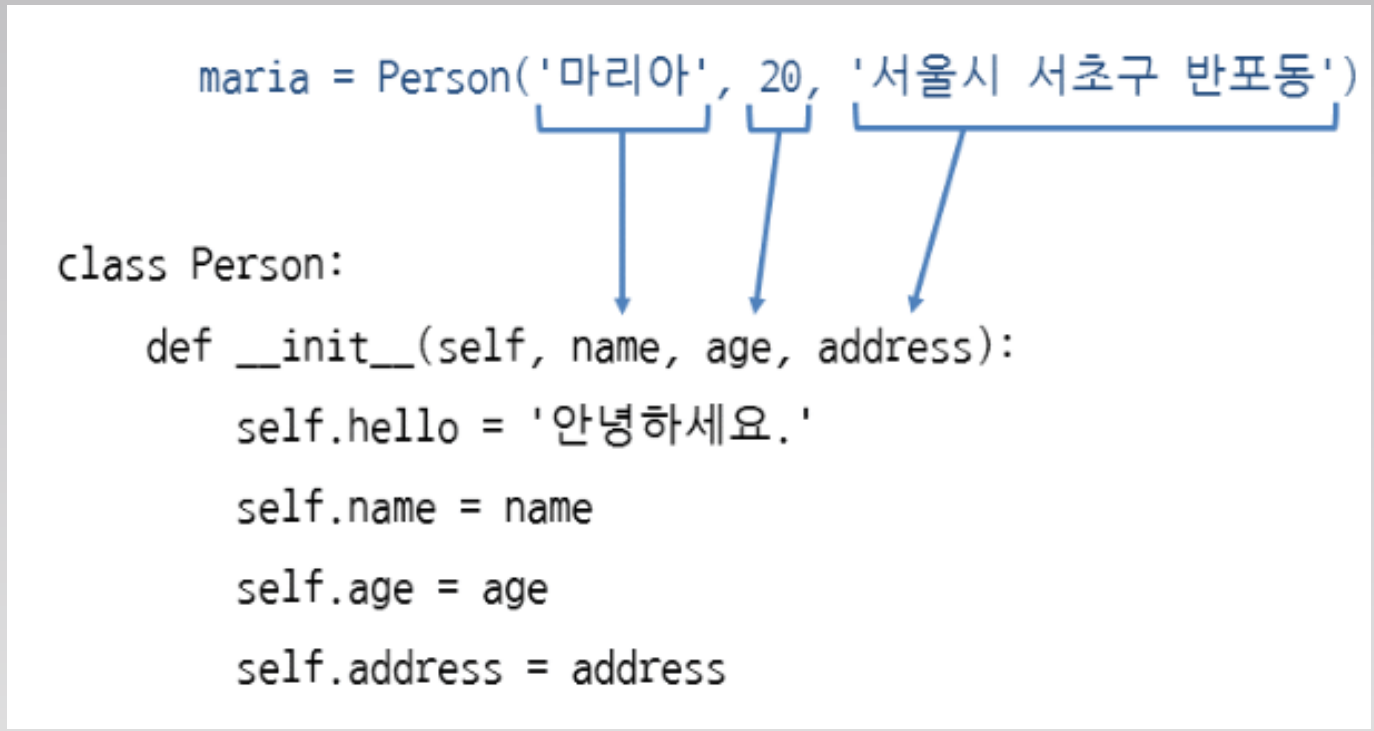
```
    def __init__(self, name, age, address):
        self.hello = '안녕하세요.'
        self.name = name
        self.age = age
        self.address = address
```

```
    def greeting(self):
        print('{0} 저는 {1}입니다.'.format(self.hello, self.name))
```

```
maria = Person('마리아', 20, '서울시 서초구 반포동')
```

34.2 속성 사용하기

▼ 그림 클래스로 인스턴스를 만들 때 값 받기



34.2 속성 사용하기

인스턴스를 만들 때 값 받기

- 클래스 바깥에서 속성에 접근할 때는 인스턴스.속성 형식으로 접근함
- 다음과 같이 maria.name, maria.age, maria.address의 값을 출력해보면 Person으로 인스턴스를 만들 때 넣었던 값이 출력됨

```
print('이름:', maria.name)       # 마리아
print('나이:', maria.age)        # 20
print('주소:', maria.address)    # 서울시 서초구 반포동
```

- 인스턴스를 통해 접근하는 속성을 인스턴스 속성이라 부름

34.3 비공개 속성 사용하기

비공개 속성 사용하기

- Person 클래스에는 hello, name, age, address 속성이 있었음

```
class Person:
    def __init__(self, name, age, address):
        self.hello = '안녕하세요.'
        self.name = name
        self.age = age
        self.address = address
```

- 이 속성들은 메서드에서 self로 접근할 수 있고, 인스턴스.속성 형식으로 클래스 바깥에서도 접근할 수 있음

```
>>> maria = Person('마리아', 20, '서울시 서초구 반포동')
>>> maria.name
'마리아'
```

34.3 비공개 속성 사용하기

비공개 속성 사용하기

- 비공개 속성은 __속성과 같이 이름이 __(밑줄 두 개)로 시작해야 함
- 단, __속성__처럼 밑줄 두 개가 양 옆에 왔을 때는 비공개 속성이 아니므로 주의해야 함

```
class 클래스이름:
    def __init__(self, 매개변수)
        self.__속성 = 값
```

34.3 비공개 속성 사용하기

비공개 속성 사용하기

- 그럼 Person 클래스에 지갑 속성 __wallet을 넣어보자

class_private_attribute_error.py

```
class Person:
    def __init__(self, name, age, address, wallet):
        self.name = name
        self.age = age
        self.address = address
        self.__wallet = wallet    # 변수 앞에 __를 붙여서 비공개 속성으로 만듦

maria = Person('마리아', 20, '서울시 서초구 반포동', 10000)
maria.__wallet -= 10000    # 클래스 바깥에서 비공개 속성에 접근하면 에러가 발생함
```

실행 결과

```
Traceback (most recent call last):
  File "C:\project\class_private_attribute_error.py", line 9, in <module>
    maria.__wallet -= 10000    # 클래스 바깥에서 비공개 속성에 접근하면 에러가 발생함
AttributeError: 'Person' object has no attribute '__wallet'
```

34.3 비공개 속성 사용하기

비공개 속성 사용하기

- 비공개 속성은 클래스 안의 메서드에서만 접근할 수 있음

class_private_attribute.py

```
class Person:
    def __init__(self, name, age, address, wallet):
        self.name = name
        self.age = age
        self.address = address
        self.__wallet = wallet    # 변수 앞에 __를 붙여서 비공개 속성으로 만듦

    def pay(self, amount):
        self.__wallet -= amount   # 비공개 속성은 클래스 안의 메서드에서만 접근할 수 있음
        print('이제 {0}원 남았네요.'.format(self.__wallet))

maria = Person('마리아', 20, '서울시 서초구 반포동', 10000)
maria.pay(3000)
```

실행 결과

```
이제 7000원 남았네요.
```

34.3 비공개 속성 사용하기

비공개 속성 사용하기

- 다음과 같이 지갑에 든 돈이 얼마인지 확인하고 돈이 모자라면 쓰지 못하는 식으로 만듦

```
def pay(self, amount):
    if amount > self.__wallet:    # 사용하려고 하는 금액보다 지갑에 든 돈이 적을 때
        print('돈이 모자라네...')
        return
    self.__wallet -= amount
```

- 중요한 값인데 바깥에서 함부로 바꾸면 안될 때 비공개 속성을 주로 사용함
- 비공개 속성을 바꾸는 경우는 클래스의 메서드로 한정함