파이썬 프로그래밍

9. 클래스(1)

00

Contents

❖ 수업 목표

- 클래스 문법을 알고 정의할 수 있다.
- 클래스 구성 맴버를 알고 코드로 작성할 수 있다.
- 객체를 생성하고 사용할 수 있다.
- self의 활용 목적을 이해하고 사용할 수 있다.
- __init__() 메서드를 사용하여 객체를 초기화 할 수 있다.

❖ 세부 목표

00

1. 클래스 구성 및 접근

❖ 절차적 프로그래밍(Procedural Programming)

- 작업의 기능적/순차적인 처리에 중점을 둔 프로그래밍
  - 장점 : 컴퓨터의 작업 처리 방식과 유사하며 OOP에 비해 상대적으로 처리속도가 빠름
  - 단점 : 유지보수가 어렵고, 코드 순서가 바뀌면 동일 결과를 보장할 수 없음

❖ 객체 지향 프로그래밍(Object Oriented Programming)

- 실제 세계의 객체를 모델로 프로그래밍
- 객체의 속성(데이터)과 동작(기능)을 묶어서 클래스로 제작
  - 장점 : 유지보수 및 코드 재사용 용이
  - 단점 : 메모리 사용이 크고 PP에 비해 속도가 상대적으로 느림

00

1. 클래스 구성 및 접근

❖ 클래스 (class)

- 분류(종류에 따라 가르는 것), 틀, 설계도
  - 예시) 위인 – 세종대왕, 신사임당, 율곡이이, 퇴계이황
  - 자동차 – 내차, 아빠차, 친구차
  - 가수 – 아이유, 김동률, BTS, 오마이걸
- 데이터(변수)와 기능(함수)를 모아 놓은 것
- 객체지향의 가장 기본적 개념

❖ 객체 (object)

- 분류에 따라 구분되는 실체, 실제 존재하는 것(클래스 선언된 식별자)
  - 예시) 위인 – 세종대왕, 신사임당, 율곡이이, 퇴계이황
  - 자동차 – 내차, 아빠차, 친구차
  - 가수 – 아이유, 김동률, BTS, 오마이걸
- 인스턴스 (instance)
  - 특정 클래스를 통해 생성된 객체, 해당 클래스의 실체(메모리 적재)

00

1. 클래스 구성 및 접근

❖ 클래스 기본 구조

- **class** 키워드 사용
- 클래스 이름은 대문자로 시작
- 클래스 이름 뒤에는 반드시 콜론(:)이 위치
- 들여쓰기가 적용된 부분까지가 클래스의 정의 부분

syntax : 클래스 정의

```
class 클래스 이름:
    def 함수이름(self):
        self.변수
```

```
class Pizzaclass:
    def order(self):
        명령문
        self.kind=10
```

1. 클래스 구성 및 접근

❖ 클래스 기본 구조

- 구성 요소
  - 맴버 변수 (Member Variable)
    - 클래스(인스턴스)의 속성, 명사로 표현
  - 맴버 함수 (Method, 메서드)
    - 클래스의 동작을 구현한 함수, 동사로 표현
    - 매개변수로 self가 포함된 함수

```
class ClassName:
    memberVariable = "속성_값"

    def methodName(self):
        """동작_코드블록"""
```

```
class 클래스:
    맴버변수1 = 속성_값1
    def 메서드1(self):
        동작_코드블록
    def 메서드2(self):
        동작_코드블록
        self.맴버변수2 = 속성_값2
```

2. 클래스(class)와 인스턴스

❖ 객체 생성

- 클래스를 통해 객체 생성
  - 변수 생성과 동일한 방식으로 객체 생성
  - 인스턴스: 특정 클래스를 통해 생성된 객체(메모리 적재 상태)

- 객체를 생성하면 객체 내에 맴버 변수와 맴버 함수가 같이 생성
  - 인수가 없는 경우, 생략 가능

syntax : 객체 생성

객체 이름 = 클래스 이름(인수)	na = Pizzaclass()

3. 객체의 맴버 접근

❖ 객체의 맴버 접근

- 객체 멤버(멤버변수, 멤버함수)의 활용을 위해 접근연산자(.) 이용
- 객체(인스턴스) 사용 방법
  - 객체.맴버변수
  - 객체.메서드()

```
class Singer:
    name = "아이유"
    def sing(self):
        print("이 밤 그날의 반딧불을 당신의 창 가까이 보낼게요~")


iu = Singer()
print(iu.name)
iu.sing()
```

3. 객체의 맴버 접근

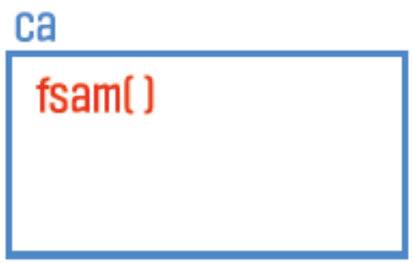
❖ 객체의 멤버 접근

▪ 클래스 정의 코드 (1)

- 멤버 함수(메소드) 정의

```
class Cexm: # ---------> ①
    def fsam(self): # ---> ②
        print("멤버 함수(메소드)")

ca=Cexm() # --------> ③
ca.fsam() # ----------> ④
```

〈객체 생성 후 객체 내부〉



00

3. 객체의 맴버 접근

❖ 객체의 멤버 접근

- 클래스 정의 코드 (2)
  - 멤버 함수(메소드)와
  - 멤버 변수를 정의
    - (인스턴스 변수)

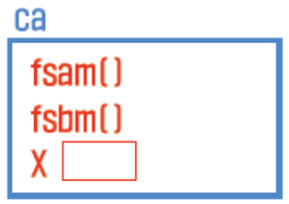
```
class Cexm:
    def fsam(self):
        print("멤버 함수(메소드)")
    def fsbm(self, pa): # ----------------> ①
        self.x=pa # --------------------> ②
        print("멤버 변수 x는", self.x)  # -> ③

ca=Cexm() # ---------------------------> ④
ca.fsam() # ---------------------------> ⑤
ca.fsbm(10) # -------------------------> ⑥
```

〈화면 출력〉

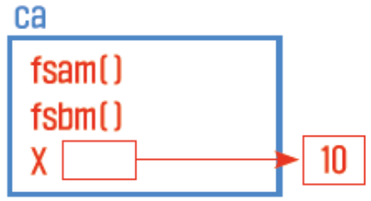
```
멤버 함수(메소드)
멤버 변수 x는 10
```

〈객체 생성 후 객체 내부〉

④ ca 객체 생성 후의 객체 내부



⑥ 실행 후의 ca 객체 내부



3. 객체의 맴버 접근

❖ 클래스 멤버와 인스턴스 멤버

- ① **클래스 멤버(변수)**: 클래스 내, 클래스 메소드 외부에서 선언
  - 선언 방법: 일반 변수와 동일, "변수명 = 값"
  - 클래스 안에 공간이 할당된 변수
  - 클래스 변수 사용 방법: "클래스 이름. 클래스 변수명"

  - 예) name = "아이유"

- ② **인스턴스 멤버(변수)**: 클래스 내, 클래스 메소드 내부에서 선언
  - 선언 방법: "self.변수명 = 값"
  - 인스턴스를 생성해야 사용할 수 있는 변수
  - 각 인스턴스에 개별적으로 변수가 생성

  - 예) self.name = "아이유"

4. self

❖ self

- 해당 클래스의 객체를 의미(해당 객체의 저장 위치 정보 포함)
- 클래스 내 메서드의 첫번째 매개변수로 사용
- 클래스를 통해 생성된 객체를 구분하기 위해 사용
- self 객체를 통해 변수와 메서드를 호출 가능

```
class Bag:
    def __init__(self):
        self.data = []

    def add(self, x):
        self.data.append(x)

    def addtwice(self, x):
        self.add(x)
        self.add(x)
```

4. self

- 문제에 제시된 프로그램이 오류 없이 제시된 화면 출력 결과가 나오도록 프로그램 일부를 변경하세요.

```
class Cadd:
    def fadd(self, a, b):
        self.x=a
        self.y=b
        self.hap=self.x+self.y

obj=Cadd()
obj.fadd(10, 20)
print("객체 obj 내의 인스턴스 변수 x의 값은", obj.x)
print("객체 obj 내의 인스턴스 변수 y의 값은", obj.y)
print("객체 obj 내의 인스턴스 변수 hap의 값은", obj.hap)
```

〈화면 출력〉

```
객체 obj 내의 인스턴스 변수 x의 값은 10
객체 obj 내의 인스턴스 변수 y의 값은 20
객체 obj 내의 인스턴스 변수 hap의 값은 30
>>>
```

5. __init__() 초기화 메서드

❖ 생성자

- 객체 생성 메서드

```
class 이름:
    def __init__(self, 초기값):
        멤버 초기화
    메서드 정의
```

- __init__() 생성자
  - 객체를 생성할 때, 초기화 역할을 수행하며 자동으로 호출되는 함수

human

```
class Human:
    def __init__(self, age, name):
        self.age = age
        self.name = name
    def intro(self):
        print(str(self.age) + "살 " + self.name + "입니다.")
kim = Human(29, "김상형")
kim.intro()
lee = Human(45, "이승우")
lee.intro()
```

실행결과

```
29살 김상형입니다.
45살 이승우입니다.
```

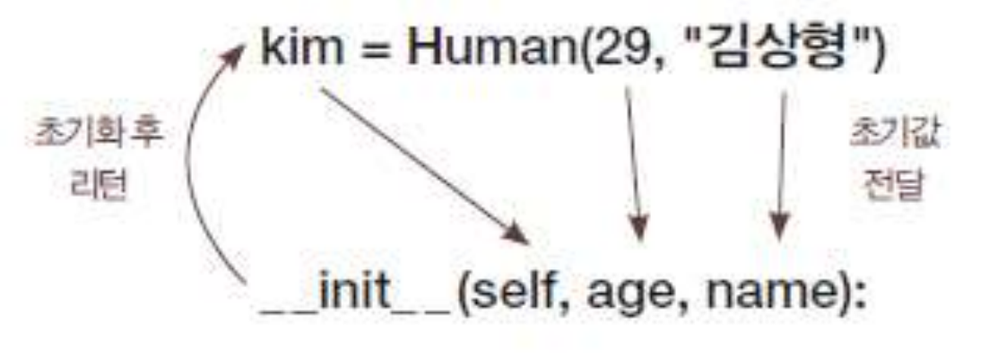
00

5. __init__() 초기화 메서드

객체 생성 구문

객체 = 클래스명(인수)

- 객체를 __init__()의 첫 번째 인수 self로 전달
- 생성문에서 전달한 인수를 두 번째 이후의 매개변수로 전달
  - 매개변수의 개수에 따라 인수의 개수 사용
- 새로 생성되는 객체 멤버에 대입



5. __init__() 초기화 메서드

❖ __init__() 메소드 예제

- **Car 클래스 정의**
  - 차량 번호와 속도를 입력 받아 초기화하는 init 메소드 작성하기

```
class Car:
    def __init__(self, pnum, pspeed):
        self.num=pnum
        self.speed=pspeed

new_car=Car(2023, 90)
print("차량 번호", new_car.num) # ----> ①
print("속도는", new_car.speed) # -----> ②
```

〈화면 출력〉

```
차량 번호 2023
속도는 90
```

5. __init__() 초기화 메서드

❖ __init__() 메소드 예제

▪ fprint 메소드 생성

- 차량번호와 속도를 화면에 출력하는 기능의 메서드 fprint 정의하고 객체 생성 및 사용하기

```
class Car:
    def __init__(self, pnum, pspeed): # ----> ①
        self.num=pnum
        self.speed=pspeed
    def fprint(self):
        print("차량 번호", self.num)
        print("속도는", self.speed)

new_car=Car(2023, 90) # ----> ②
new_car.fprint()
```

〈화면 출력〉

```
차량 번호 2023
속도는 90
```

00

6. 내장 클래스

❖ 내장 클래스(Built-In Class)

- 파이썬 내부에 기 정의된 클래스
- 클래스가 객체(Object)를 생성하는 것
  : 인스턴스화(Instantiate)

syntax : 정확한 코딩

`na = 10`	`na = int(10)`

syntax : 변수와 객체 생성 설명

- na = 10은 10이라는 정수 객체를 생성하고 이 정수 객체를 참조할 수 있는 정보(주소 data)를 na 변수에 할당하는 과정이다.
- 할당을 파이썬으로 해석하면 int 객체 10을 생성하고 10을 참조할 수 있는 정보를 na 변수에 저장하는 것이다. '참조할 수 있는 정보'는 '객체 10'이 저장된 메모리 주소(주소 data)이고 na 변수에는 그 주소 data가 할당되어 저장되었다고 이해하는 것이 좋다.
- int는 클래스(템플릿)이다.
- na = int(10) 이 변수(객체) 생성 과정이다.
- na는 객체명(변수명)이다. int는 클래스(Built-in class)로 매개변수 10을 넣어 객체를 생성하고 그 객체에 na이라고 이름을 붙이는 것이다. int 클래스에 매개변수가 하나 있는 __init__(self, pa) 초기화 메소드가 정의되어 있어 int(10)으로 객체를 생성할 때 자동으로 호출되는 것이라고 추측할 수 있을 것이다.

6. 내장 클래스

❖ 내장 클래스(Built-In Class)

- 예시 코드
- str 클래스의 객체
  - 우측 코드에서 str 클래스의 객체를 생성하면서 문자열 "abc"를 인수로 받는 __init __(self, pa) 메소드 호출

```
stra="abc"
print(stra)
```

```
strb=str("abc")
print(strb)
strc=strb.capitalize() # -----> ①
print(strc)
```

6. 내장 클래스

❖ 내장 클래스와 사용자 정의 클래스

- **파이썬에서 정의한 클래스(파이썬에서 제공하는 데이터형)** : int, float, str, list, dictionary
- **사용자가 정의한 클래스** : 사용자가 클래스를 정의하고 이름을 붙인다.

00

7. 연습 문제

- 디폴트 매개변수를 이용하여 제시된 화면 출력 결과가 나오도록 fadd 함수를 정의하세요.

```
ohap=fadd(10, 20)
print("ohap의 값은", ohap)
ohap2=fadd(10)
print("ohap2의 값은", ohap2)
ohap3=fadd()
print("ohap2의 값은", ohap3)
```

〈화면 출력〉

```
ohap의 값은 30
ohap2의 값은 12
ohap2의 값은 3
>>>
```

00

7. 연습 문제

- 디폴트 매개변수의 사용을 응용하여 클래스를 변경해 보세요.

```
class Cadd:
    (                                    )
        self.x=a
        self.y=b
        self.hap=self.x+self.y
obj=Cadd()
obj.fadd(10, 20)
print("객체 obj 내의  인스턴스 변수  hap의 값은", obj.hap)
obj.fadd(10)
print("객체 obj 내의  인스턴스 변수  hap의 값은", obj.hap)
obj.fadd()
print("객체 obj 내의  인스턴스 변수  hap의 값은", obj.hap)
```

〈화면 출력〉

```
객체 obj 내의  인스턴스 변수  hap의 값은 30
객체 obj 내의  인스턴스 변수  hap의 값은 12
객체 obj 내의  인스턴스 변수  hap의 값은 3
```

00

7. 연습 문제

- 다음과 같은 속성과 동작을 포함하는 동물(Animal) 클래스를 생성하고, 고양이(cat) 객체를 생성하여 속성을 출력하고 동작을 수행하시오.
  - 속성: 이름(name)= ‘고양이’
  - 동작: ‘야옹’ 소리내다(sound)

00

7. 연습 문제

- 다음과 같은 속성과 동작을 포함하는 과일(Fruit) 클래스를 생성하고, 오렌지(orange) 객체를 생성하여 속성을 출력하고 동작을 수행하시오.
  - 속성: 이름(name)= ‘오렌지’ , 색상(color)= ‘노란색’
  - 동작: ‘새콤하다’ 맛이나다(taste)

00

Notice

❖ 과제

- 1. 객체지향 프로그래밍과 클래스에 대해 설명하기
- 2. 클래스 구성 요소 및 문법 설명하기
- 3. 객체 생성 및 사용 코드 작성하기
- 4. self 키워드 설명하고 사용 방법 확인하기
- 5. __init__() 매서드 사용하여 객체 초기화 하기
- 6. 내장 클래스와 사용자 정의 클래스 구분하기

❖ 다음 수업 내용

- 클래스
  - 모듈, 맴버 변수, from import, 모듈 구조화, 상속, 오버라이딩