파이썬 프로그래밍

10. 클래스(2)

00 Contents

❖ 수업 목표

- 메인 및 하위 모듈을 생성하고 동작 방식을 이해 할 수 있다.
- 맴버 변수를 활용할 수 있다.
- 3가지 import 형태를 사용할 수 있다.
- 오버라이딩을 포함한 상속 예제 코드를 작성할 수 있다.

❖ 세부 목표

00

1. 모듈 생성

❖ 모듈

- 클래스, 함수, 변수 등을 모아둔 파일
  - 특정 기능을 하는 프로그램들을 모아둔 파일
- 파이썬 파일(.py)
- 프로그램 만드는 데 드는 노력 절감

1. 모듈 생성

❖ 모듈 생성

- 리스트를 맴버 변수로 가지는 클래스 정의
  - add() 함수와 fprint() 함수를 메서드로 포함
  - 모든 데이터형은 클래스의 멤버변수로 사용 가능

```
class Cvalue:

    def __init__(self):
        self.lista=[] # ----------------------> ①
    def add(self,num):
        self.lista.append(num) # -------------> ②
    def fprint(self):
        print(self.lista)
p1=Cvalue() # -------------> ③
p1.add(1) # -------------> ④
p1.add(2)
p1.add(3)
p1.fprint()
```

p1 객체 내부

- <화면 출력 결과>
  - [1, 2, 3]

00

1. 모듈 생성

❖ 모듈 생성

- plus() 함수 추가
- 프로그램을 파일명 mex1.py 로 저장 => mex1 모듈 생성 완료

'mex1.py' 파일

```
class Cvalue:
    def __init__(self):
        self.lista=[]
    def add(self,num):
        self.lista.append(num)
    def fprint(self):
        print(self.lista)
def plus(a, b): # ------------------> ①
    c= a + b
    return c

p1=Cvalue()
p1.add(1)
p1.add(2)
p1.add(3)
p1.fprint()
```

2. import

❖ import

- 모듈 불러오는 키워드

syntax : 정확한 코딩

import 모듈 이름	import mex1

2. import

❖ import

- 불러온 모듈 뒤에 접근연산자(.)를 붙여 해당 모듈이 포함한 내용 사용
  - mex1 프로그램도 자동 실행 됨

```
import mex1

p2=mex1.Cvalue()
p2.add(11)
p2.add(21)
p2.add(31)
p2.fprint()
value=mex1.plus(10, 20) # --> ②
print(value)
```

〈화면 출력〉

```
[1, 2, 3] # ----------------> ①
[11, 21, 31]
30
```

3. from 모듈명 import 함수

❖ from 모듈명 import 함수

- import 한 모듈의 이름을 쓰고 접근 연산자를 사용하는 것이 복잡한 경우 사용
- 다음 형식과 같이 import 하면 간편하게 사용 가능

syntax : 정확한 코딩

from 모듈 이름 import 함수	from mex1 import plus

3. from 모듈명 import 함수

❖ from 모듈명 import 함수 코드

- 모듈(라이브러리)을 import 시, 전체 모듈이 아닌 필요한 부분만 사용 가능
  - 단, 이 경우도 mex1 프로그램이 자동 실행 됨

```
from mex1 import plus

value=plus(10, 20)
print(value)
```

```
[1, 2, 3] # ----------------> ①
30
```

3. from 모듈명 import 함수

❖ from 모듈명 import 함수 코드

- 다음과 같이 import * 형식으로 모듈 이름과 접근 연산자 없이 모든 내용을 사용 가능
  - *는 프로그램에서 "모든 것"을 의미하는 기호
- 다만, 이 방식은 코드의 가독성을 나쁘게 하며, 현재 영역에서 정의된 객체 이름이 충돌할 수 있으므로 권장하지 않음

syntax : 정확한 코딩

from 모듈 이름 import *	from mex1 import *

4. 메인 모듈과 하위 모듈

❖ 메인 모듈과 하위 모듈

- 메인 모듈 : 제일 먼저 실행되는 모듈
- 하위 모듈 : 메인 모듈이 import 문을 이용하여 불러오는 모듈

- 추가 코드 작성
  - ‘mex1.py’ 프로그램에 ① 부분을 추가 코딩하고 ‘mex4.py’ 로 저장

```
class Cvalue:
    def __init__(self):
        self.lista=[]
    def add(self.num):
        self.lista.append(num)
    def fprint(self):
        print(self.lista)

def plus(a, b)
    c=a+b
    return c

if __name__=="__main__":  # ------> ①
    p1=Cvalue()
    p1.add(1)
    p1.add(2)
    p1.add(3)
    p1.fprint()
```

② 콜론

③ 들여쓰기

〈화면 출력〉

```
[1, 2, 3]
```

4. 메인 모듈과 하위 모듈

❖ 메인 모듈과 하위 모듈

- __name__ 내장 변수
  - 메인 모듈의 경우, __main__ 이 저장됨
  - 하위 모듈의 경우, 모듈명이 저장됨

- 예시 코드 작성하기 : mex5.py

```
from mex4 import plus

value=plus(10, 20)
print(value)
```

〈화면 출력〉

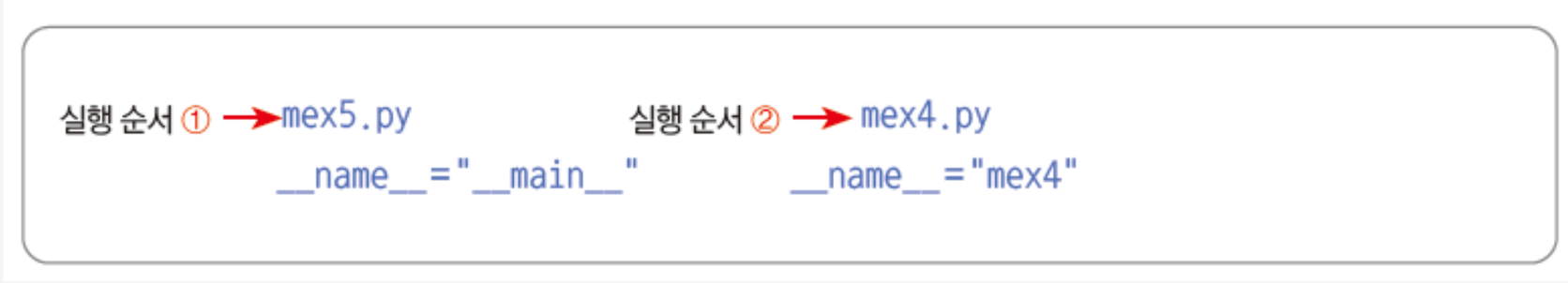
```
30
```

4. 메인 모듈과 하위 모듈

❖ 메인 모듈과 하위 모듈

- mex5.py 실행 시, 다음과 같이 구성됨
  - 메인 모듈 : mex5.py
  - 하위 모듈 : mex4.py

- 하위 모듈의 __name__에 mex4 가 저장되므로 if문은 실행되지 않음
  - 즉, mex5.py에서 실행한 내용만 화면에 출력됨



4. 메인 모듈과 하위 모듈

❖ library 모듈

- 내장 라이브러리인 turtle 모듈 import 하여 사용하기
  - 내장 라이브러리: 파이썬 제공 모듈 모음

- 파이썬 turtle 모듈 설명 페이
  - https://docs.python.org/3/library/turtle.html 를 참조

4. 메인 모듈과 하위 모듈

❖ random 모듈

코드 8-6 하나의 값을 임의로 선택하는 코드

```
01:  import random
02:  animals = ['체셔고양이', '오리', '도도새']
03:  print(random.choice(animals))
```

↳ 실행화면

```
도도새
```

- **random.choice(리스트)**
  - 리스트의 값 중 하나를 임의로 선택
- **random.sample(리스트, 뽑을_개수)**
  - 리스트의 값 중에서 지정한 개수만큼 중복 없이 임의로 선택
- **random.randint(시작_값, 끝_값)**
  - 지정한 범위에서 정수 하나를 임의로 선택

5. 상속

❖ 상속(inheritance)

- 어떤 클래스가 다른 클래스의 성질을 물려받는 것
  - 기존 클래스를 확장하여 멤버 추가하거나 동작 변경
  - 자식 클래스 이름 옆 괄호 안에 부모 클래스 이름 지정

```
class 이름(부모):
    ....
```

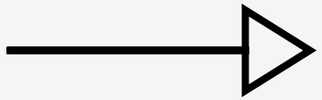
  - 부모클래스(super class)
  - 자식클래스(sub class)

5. 상속

❖ 상속(inheritance)

- 자식 클래스는 부모 클래스를 상속
  - 예) 학생 클래스는 인간 클래스 상속

```
class Student(Human):
    def study(self):
        print("공부하다")
```



```
class Human:
    eyes = 2
    nose = 1
    mouth = 1

    def __init__(self, name, age):
        self.name = name
        self.age = age

    def introduce(self):
        print(f"이름:{self.name}")
        print(f"나이:{self.age}")

    def eat(self):
        print('먹다')

    def sleep(self):
        print('자다')

    def talk(self):
        print('말하다')
```

6. 오버라이딩

❖ 오버라이딩

- 부모 클래스의 메소드를 동일한 이름으로 자식 클래스에서 재정의하여 사용하는 것
  - 부모 메서드가 자식 클래스에서 재정의 되지 않으면 그대로 사용
  - 부모 메서드가 자식 클래스에서 재정의 되면 자식 메서드를 실행

```
class Animal:
    def eat(self):
        print("먹다")
    def move(self):
        print("이동하다")
    def sound(self):
        print("소리내다")

class Cat(Animal):
    def eat(self):
        print("잡식하다")
    def sound(self):
        print("야옹~ ", end="")
        super().sound()

cat1 = Cat()
cat1.eat()
cat1.move()
cat1.sound()
```

- 실행 결과

```
잡식하다
이동하다
야옹~ 소리내다
```

6. 오버라이딩

- **super() 클래스**
  - 자식 클래스에서 부모의 메서드 호출할 때 사용
  - 부모 클래스를 의미

```
class Student(Human):
    def __init__(self, name, age, studentNum):
        super().__init__(name, age)
        self.studentNum = studentNum

    def introduce(self):
        super().introduce()
        print(f"학번:{self.studentNum}")

    def study(self):
        print("공부하다")


hong = Human("홍길동", 33)
hong.introduce()
kim = Student("김유빈", 20, 20220523)
kim.introduce()
kim.study()
```

7. 연습문제

❖ 중복되지 않는 카드 두 장을 뽑도록 빈칸을 채워보세요

```
01:  import random
02:  clovers = ['클로버1', '클로버2', '클로버3']
03:  print(                          )
```

7. 연습문제

❖ 다음 코드가 동작하도록 자동차 클래스를 정의하세요.

- >> car = Car(4, 3000)
- >> print(car.wheel)
- 4
- >> print(car.price)
- 3000

00

7. 연습문제

- 자동차(Car) 클래스를 상속받은 자전거(Bicycle) 클래스를 정의하세요.

7. 연습문제

❖ 다음 코드가 동작하도록 자전거 클래스를 정의하세요.

- >> bicycle = Bicycle(2021, 2, 100)
- >> print(bicycle.year)
- 2021
- >> print(bicycle.wheel)
- 2
- >> print(bicycle.price)
- 100

7. 연습문제

- 자동차 클래스를 상속받아 다음 코드가 동작하도록 자전거 클래스를 정의하세요.
  - >> bicycle = Bicycle(2021, 2, 100, "시마노")
  - >> bicycle.drivetrain
  - 시마노

7. 연습문제

❖ 다음 코드가 동작하도록 자전거 클래스를 수정하세요.

- >> bicycle = Bicycle(2021, 2, 100, "시마노")
- >> bicycle.info()
- year : 2021
- wheel : 2
- price : 100

7. 연습문제

❖ 자전거의 info() 매서드로 drivetrain 정보까지 출력하도록 수정하세요
.

- >> bicycle = Bicycle(2021, 2, 100, "시마노")
- >> bicycle.info()
- year : 2021
- wheel : 2
- price : 100
- drivetrain : 시마노

00

Notice

❖ 과제

- 1. 모듈 생성 및 동작 코드 작성하기
- 2. 맴버 변수 생성 및 동작 방식 확인하기
- 3. 모듈 import 방식 다양하게 사용하기
- 4. __name__ 변수와 함께 메인 및 하위 모듈 사용하기
- 5. 상속 코드 작성하고 오버라이딩 코드 동작사항 이해하기

❖ 다음 수업 내용

- tkinter
  - 위젯 생성, Geometry manager, 이벤트, 바인딩