파이썬 프로그래밍

11-1: 클래스

2022.5.10

인하대학교 INHA UNIVERSITY 산업경영공학과 임세실

Overview

- 객체지향 프로그래밍 이해하기
- Class 이해하기
- 클래스의 고급사용

객체지향 프로그래밍 이해하기

파이썬을 기능별로 나눠서 프로그래밍하는 방법

- 함수
- 모듈
- 객체지향 프로그래밍 (Object Oriented Programming)

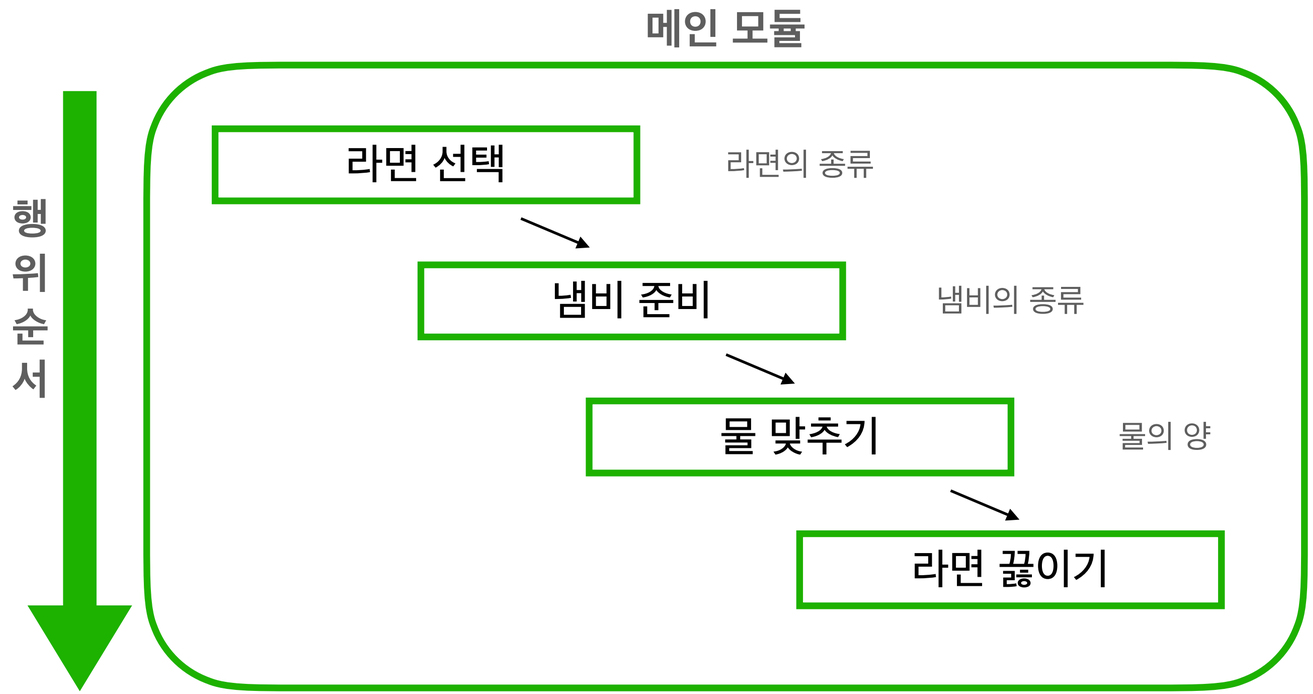
객체지향 프로그래밍이란? (1)

- 우리가 여태까지 해온 방식: **절차지향적 프로그래밍!**

절차지향적 프로그래밍

- 행위순에 따라 코드 작성

라면 끓이기의 절차지향적 프로그래밍



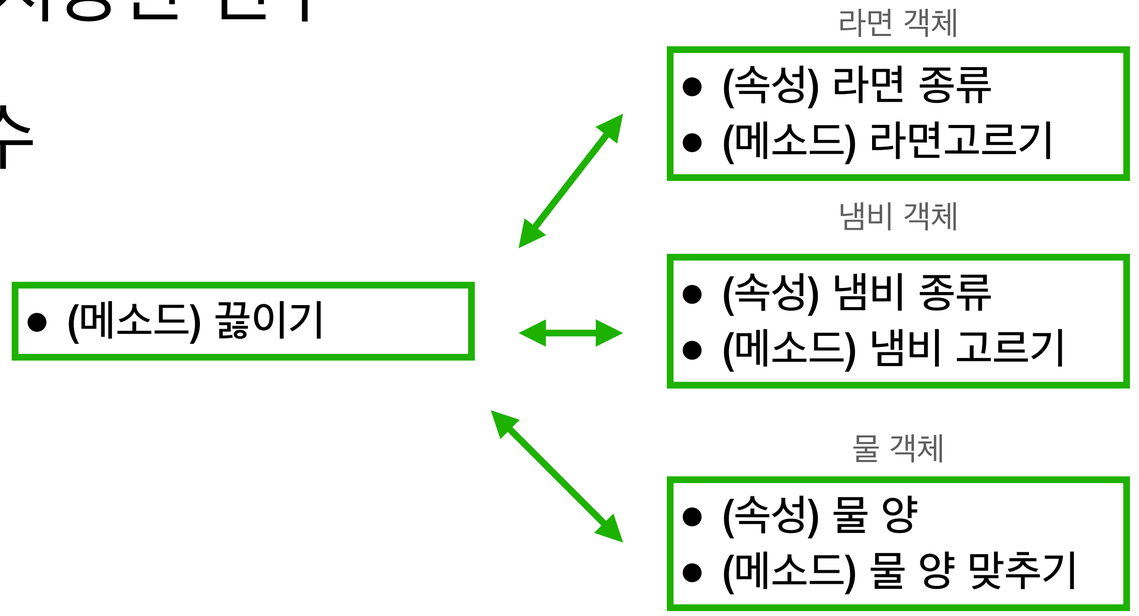
객체지향 프로그래밍이란? (2)

- 프로그램의 구성요소들을 독립된 객체(object)로 정의

객체 = 속성 (Attribute) + 기능 (Method)

  - 속성: 관심대상의 특징 및 데이터를 저장한 변수
  - 기능: 관심대상의 행위를 묘사한 함수

라면 끓이기의 객체지향적 프로그래밍



객체지향 프로그래밍을 왜 하는 걸까?

- 프로그램 내부 요소의 독립성 확보
- 지속적 수정 및 업그레이드의 용이성
- 분업 용이성

Class 이해하기

- 클래스 정의하기
- 생성자란?
- 메소드 정의하기
- 인스턴스 생성하기
- 메소드 사용하기

학생 성적 관리 프로그램을 객체지향 프로그램으로 구현

- 학생별로 국어, 수학, 영어, 과학 성적 데이터를 가지고 있음
- 각 학생들의 성적 총점과 평균 점수를 구해 출력하는 프로그램 작성
- 학생 객체가 다음과 같은 속성 및 기능 가짐
  - 속성(Attribute): 학생이름, 국어점수, 수학점수, 영어점수, 과학점수
  - 기능(Method): 총점 구하기, 평균 점수 구하기, 결과 출력하기

➡ 클래스 (class) 사용해 객체의 효율적 구현 가능!

클래스 (class)란?

- 파이썬에서 객체를 어떻게 구성할 것인에 대한 설계도
- 파이썬의 자료형의 하나
- 설계도(클래스)를 바탕으로 실제로 구현된 객체를 인스턴스 (instance)라고 부름

클래스 정의하기

```
class 클래스이름:
    클래스내용
```

- 클래스이름
  - 대문자로 시작
  - 목적과 의미가 명확해야 함
  - 영문 대소문자, 숫자, _로 구성
  - 숫자로 시작 불가
- 클래스내용
  - 클래스의 멤버인 변수와 메소드(클래스 안의 함수) 선언

Student 클래스 정의하기

```
class Student:
    def __init__(self, name, korean, math, english, science):
        self.name = name
        self.korean = korean
        self.math = math
        self.english = english
        self.science = science

    def get_sum(self):
        return self.korean + self.math + self.english + self.science

    def get_average(self):
        return self.get_sum()/4

    def get_string(self):
        return "{}\t{}\t{}".format(self.name,self.get_sum(),self.get_average())
```

➡생성자(Constructor)

➡메소드

생성자란?

- 객체 생성 시 변수 선언 및 초기화 담당하는 메소드
- 객체 자기 자신을 의미하는 self를 매개변수로 받음

```
def __init__(self, name, korean, math, english, science):
    self.name = name
    self.korean = korean
    self.math = math
    self.english = english
    self.science = science
```

메소드 정의하기

- 일반적인 함수 선언하는 방법과 동일함
- 메소드마다 self를 반드시 첫번째 매개변수로 선언해야함

```
def get_sum(self):
    return self.korean + self.math + self.english + self.science

def get_average(self):
    return self.get_sum()/4

def get_string(self):
    return "{}\t{}\t{}".format(self.name,self.get_sum(),self.get_average())
```

인스턴스 (객체) 생성하기

- 클래스 이름과 똑같은 생성자 사용해서 인스턴스 생성함

인스턴스이름(변수이름) = 클래스이름()

- Student 인스턴스 생성

```
students = [
    Student("Andrea", 87, 98, 88, 95),
    Student("Betty", 37, 48, 28, 55),
    Student("Charlie", 73, 52, 74, 53),
    Student("Dorothy", 85, 99, 96, 97),
    Student("Gerhard", 68, 86, 65, 66),
    Student("Homns", 82, 95, 83, 95)
    ]
```

메소드 사용하기

- 메소드가 정의된 인스턴스 이름을 점과 함께 호출

  인스턴스이름.메소드이름

- Student 인스턴스 내 get_string 메소드 사용

```
print("Name","Total", "Average")
for s in students:
    print(s.get_string())
```

✓ 0.1s

```
Name Total Average
Andrea  368     92.0
Betty   168     42.0
Charlie 252     63.0
Dorothy 377     94.25
Gerhard 285     71.25
Homns   355     88.75
```

클래스의 고급사용

- 어떤 클래스의 인스턴스인지 확인하기
- 클래스 상속하기
- 프라이빗 변수 상속하기

어떤 클래스의 인스턴스인지 확인하기 (1)

- 인스턴스가 어떤 클래스로부터 만들어졌는지 확인하기

isinstance(인스턴스, 클래스)

- 해당 클래스 기반이면 True, 아니면 False 반환

```
student=Student("A",35,25,15,6)
isinstance(student, Student)
```

[8] ✓ 0.9s

```
True
```

어떤 클래스의 인스턴스인지 확인하기 (2)

```python
class Student:
    def study(self):
        print("공부를 합니다")

class Teacher:
    def teach(self):
        print("학생을 가르칩니다")

lectureroom = [Student(), Student(), Teacher(), Student(), Student()]

for i in lectureroom:
    if isinstance(i, Student):
        i.study()
    elif isinstance(i, Teacher):
        i.teach()
```

[9] ✓ 0.1s Python

```
공부를 합니다
공부를 합니다
학생을 가르칩니다
공부를 합니다
공부를 합니다
```

클래스 상속(inheritance)하기 (1)

- 클래스 만들 때 다른 클래스의 기능 물려받기
- 기반이 되는 클래스를 부모 (parent) 클래스, 물려받아 새로 만드는 클래스를 자식 (child) 클래스라고 정의
  - 부모 클래스의 모든 변수와 메소드 사용 가능
- 클래스 정의할 때 상속 받고자 하는 클래스 이름을 괄호 안에 기입

```
class 자식클래스이름(부모클래스 이름)
```

클래스 상속(inheritance)하기 (2)

- 왜 상속을 하는가?
  - 기존 클래스의 큰 변경 없이 기능 추가하거나 수정하고자 할 때
  - 기존 클래스가 패키지 형태로만 제공하거나 수정 허용하지 않는 상황일 때

클래스 상속(inheritance)하기 (3)

- 부모 클래스: Calculator
  - 두 수의 사칙연산 계산기
  - 계산기 객체의 속성과 기능
    - 속성(Attribute): 연산에 필요한 두 수
    - 기능(Method): 덧셈, 뺄셈, 곱셈, 나눗셈

클래스 상속(inheritance)하기 (4)

- 부모 클래스: Calculator

```python
class Calculator:
    def __init__(self, a, b):
        self.a = a
        self.b = b

    def add(self):
        return self.a+self.b

    def sub(self):
        return self.a-self.b

    def mult(self):
        return self.a*self.b

    def div(self):
        return self.a/self.b
```

클래스 상속(inheritance)하기 (5)

- 자식 클래스: MoreCalculator
  - Calculator의 상속
  - 첫번째 수를 두번째 수만큼 제곱하는 기능이 추가

```
class MoreCalculator(Calculator):
    def pow(self):
        return self.a**self.b

a=MoreCalculator(4,2)

print(a.add())
print(a.pow())
```

[10] ✓ 0.5s

```
6
16
```

클래스 상속(inheritance)하기 (5)

- 자식 클래스: MoreCalculator
  - div 메소드의 수정 **(오버라이드)**

Calculator 클래스

```
    def div(self):
        return self.a/self.b

b=Calculator(4,0)

print(b.div())
```

⊗ 0.1s

```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
Untitled-1.ipynb Cell 4' in <cell line: 20>()
     16         return self.a/self.b
     18 b=Calculator(4,0)
---> 20 print(b.div())

Untitled-1.ipynb Cell 4' in Calculator.div(self)
     15 def div(self):
---> 16     return self.a/self.b

ZeroDivisionError: division by zero
```

MoreCalculator 클래스

```
class MoreCalculator(Calculator):
    def div(self):
        if self.b ==0:
            return 0
        else:
            return self.a/self.b

a=MoreCalculator(4,0)
print(a.div())
```

✓ 0.5s

```
0
```

클래스 상속(inheritance)하기 (6)

- super() 써서 자식 클래스에서 부모 클래스 객체 사용하기
  - super()는 부모 클래스를 접근할 수 있게 하는 함수
  - 자식 클래스에서 부모 클래스의 속성에 일부 속성 추가 가능
  - 자식 클래스에서 부모 클래스의 메소드에 일부 기능 덧대기 가능

클래스 상속(inheritance)하기 (7)

- super() 써서 자식 클래스에서 부모 클래스 객체 사용하기

```
class MoreCalculator(Calculator):
    def __init__(self, a,b,c):
        super().__init__(a,b)
        self.c=c

    def add(self):
        print("덧셈입니다")
        return super().add()+self.c

a=MoreCalculator(4,1,5)
print(a.add())
```

✓ 0.6s

```
덧셈입니다
10
```

프라이빗 (private) 변수 사용하기 (1)

- 객체 중 일부 변수(속성)를 남이 볼 수 없게 보호
- 프라이빗 변수로 지정하는 이유
  - 다른 사람에게 코드 전달할 경우, 본래 의도대로 프로그램이 작동할 수 있게 제한을 두어 오용을 방지
  - 필요없는 정보를 숨김
  - 제품 판매 시 소스의 보호

프라이빗 (private) 변수 사용하기 (2)

- 프라이빗 변수의 선언

  __변수이름

  - 클래스 선언 시, 클래스 내부에서 변수이름 앞에 언더바 두번 붙임

- 클래스 외부에서 접근 불가능한 변수가 됨

프라이빗 (private) 변수 사용하기 (3)

- 원의 둘레 및 넓이 구하기: 반지름을 프라이빗 변수로 선언

```python
import math

class Circle:
    def __init__(self, radius):
        self.__radius = radius

    def get_length(self):
        return 2*math.pi*self.__radius

    def get_area(self):
        return math.pi*(self.__radius**2)

circle = Circle(10)
print("둘레:",circle.get_length())
print("넓이:",circle.get_area())

print("반지름:",circle.__radius)
```

⊗ 0.1s

```
둘레: 62.83185307179586
넓이: 314.1592653589793

---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Untitled-1.ipynb Cell 6' in <cell line: 17>()
     14 print("둘레:",circle.get_length())
     15 print("넓이:",circle.get_area())
---> 17 print("반지름:",circle.__radius)

AttributeError: 'Circle' object has no attribute '__radius'
```

프라이빗 (private) 변수 사용하기 (4)

- 게터(getter): 프라이빗 변수에 접근할 수 있도록 해주는 함수

```
@property
def 프라이빗변수이름(self):
    return self.__프라이빗변수이름
```

➡@: 함수 데코레이터

- 세터(setter): 프라이빗 변수의 값을 설정해주는 함수

```
@프라이빗변수이름.setter
def 프라이빗변수이름(self,value):
   self.__프라이빗변수이름 = value
```

- 객체이름.프라이빗변수이름으로 접근 가능

프라이빗 (private) 변수 사용하기 (5)

- 원의 둘레 및 넓이 구하기: 반지름을 프라이빗 변수로 선언

```python
import math

class Circle:
    def __init__(self, radius):
        self.__radius = radius

    def get_length(self):
        return 2*math.pi*self.__radius

    def get_area(self):
        return math.pi*(self.__radius**2)

    @property
    def radius(self):
        return self.__radius

    @radius.setter
    def radius(self,value):
        self.__radius = value
```

```python
circle = Circle(10)
print("둘레:",circle.get_length())
print("넓이:",circle.get_area())
print("반지름:",circle.radius)
circle.radius = 2
print("둘레:",circle.get_length())
```

✓ 0.1s

```
둘레: 62.83185307179586
넓이: 314.1592653589793
반지름: 10
둘레: 12.566370614359172
```

실습 (5/12)

- 강의실에서 출석 확인 예정
  - 온라인 출석확인 대상자는 과제 제출 여부로 출석 확인
- 실습과제는 실습시간 정시에 iClass에 업로드될 예정
- 5월 13일 자정까지 제출해야 과제 점수 인정