Study Title _____ Date _____

Chapter8.클래스

8-1클래스의 기본

-c언어를 제외한 모든 프로그래밍 언어는 객체 지향 프로그래밍 언어입니다. 객체 지향 프로그래밍이란 객체를 우선으로 생각해서 프로그래밍한다는 의미입니다.
이 모든 프로그래밍 언어는 클래스 기반의 객체 지향 프로그래밍 언어이고, 이는 클래스라는 것을 기반으로 객체를 만들고, 그러한 객체를 우선으로 생각해서 프로그래밍한다는 것입니다.

객체

-속성을 가질 수 있는 모든 것을 객체라고 부릅니다.

#추상화

-프로그램에서 필요한 요소만을 사용해서 객체를 표현하는 것을 추상화라고 부릅니다. 조금 더 포괄적인 사전적 의미로 설명하면 복잡한자료,모듈,시스템 등으로부터 핵심적인 개념 또는 기능을 간추려 내는 것을 추상화라 합니다.

학생 성적 관리 프로그램을 만든다면 무엇이 필요할까요?
ex)

```
students = [
    {"name":"윤인성", "korean":87, "math":98},
    {"name":"연하진", "korean":92, "math":98},
    {"name":"구지연", "korean":76, "math":96},
    {"name":"나선주", "korean":98, "math":92},
    {"name":"윤아린", "korean":95, "math":98},
    {"name":"윤명월", "korean":64, "math":88},
]

print("이름", "총점", "평균", sep="\t")

for student in students:
    score_sum = student["korean"]+student["math"]
    score_average = score_sum / 2
    print(student["name"],score_sum,score_average\
    , sep="\t")
>>>
이름    총점    평균
윤인성 160    92
연하진 167    93
구지연 162    94
나선주 162    95
윤아린 168    91
윤명월 169    90
```

현재 총점과 평균을 구하는 처리는 학생을 대상으로만 이루어집니다. 따라서 학생을 매개변수로 받는 형태의 함수로 만들면 코드가 조금 더 균형잡히지 않을까요?
ex)

```
def create_student(name, korean, math):
    return {
        "name": name,
        "korean": korean,
        "math": math,
    }

def student_get_sum(student):
    return student["korean"] + student["math"]

def student_get_average(student):
    return student_get_sum(student) / 2

def student_to_string(student):
    return "{}\t{}\t{}".format(
        student["name"],
        student_get_sum(student),
        student_get_average(student))
--------이까지 학생 객체와 관련된 부분--------
students = [
    create_student("윤인성", 87, 98),
    create_student("연하진", 92, 98),
    create_student("구지연", 87, 96),
    create_student("나선주", 76 92),
    create_student("윤아린", 95, 98),
    create_student("윤명월", 64, 88)
]

print("이름", "총점", "평균", sep="\t")
for student in students:
    print(student_to_string(student))
--------객체를 활용하는 처리--------
```

실행결과는 이전과 같지만 코드가 조금 분리되었습니다 학생이라는 객체와 관련된 기능이 위로 올라갔고, 이러한 객체를 사용하는 처리가 아래로 내려갔습니다. 이렇게 만들면 학생객체와 관련된 기능을 별도의 모듈로 빼서 관리할 수도 있습니다.

이러한 형태로 객체와 관련된 코드를 분리할 수 있게 하는것이 객체 지향 프로그래밍의 핵심이라고 할 수 있습니다. 그런데 이런 코드가 너무 자주사용되다 보니, 개발자들은 좀더 코드를 효율적으로 만들기 위해 클래스라는 구조를 만들게 되었습니다.

Study Title _____ Date _____

클래스 선언하기

-클래스는 객체를 좀 더 효율적으로 생성하기 위해서 만들어진 구문입니다.

"""
class 클래스이름:
    클래스 내용
"""

이렇게 만들어진 클래스는 클래스 이름과 같은 함수(생성자)를 사용해서 객체를 만듭니다.

"""
인스턴스 이름 = 클래스 이름() <- 생성자 함수
“”

이렇게 클래스를 기반으로 만들어진 객체를 인스턴스라고 부릅니다.

생성자

-클래스 이름과 같은 함수를 생성자라고 부릅니다. 클래스 내부에 __init__라는 함수를 만들면 객체를 생성할때 처리할 내용을 작성할 수 있습니다.

"""
class 클래스 이름:
    def __init__(self, 추가적인 매개변수):
        pass
"""

클래스 내부의 함수는 첫번째 매개변수로 반드시 self를 입력해야 합니다. 이때 self는 자기자신을 나타내는 딕셔너리라고 생각하면 됩니다. 다만 self가 가지고 있는 속성과 기능에 접근할 때는 self.<식별자> 형태로 접근합니다.

ex)

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

students = [
    Student(“윤인성”, 87, 98),
    Student(“연하진”, 92, 98),
    Student(“구지연”, 87, 96),
    Student(“나선주”, 76 92),
    Student(“윤아린”, 95, 98),
    Student(“윤명월”, 64, 88)
]

students[0].name
students[0].korean
students[0].math
```

소멸자

생성자와 반대로 인스턴스가 소멸될때 호출되는 함수도 있습니다. 이를 소멸자라고 부릅니다.

ex)

“”

```
class Test:
    def __init__(self, name):
        self.name = name
        print(“{} - 생성되었습니다.”.format(self.name))
    def __del__(self)
        print(“{} - 파괴되었습니다.”.format(self.name))
test = Test(“A”)
>>
A - 생성되었습니다.
A - 파괴되었습니다.
"""
```

코드를 실행하면 Test(“A”)가 실행될때 생성자가 호출되며, 프로그램이 종료될때 소멸자가 호출됩니다.

메소드

클래스가 가지고 있는 함수를 메소드라고 부릅니다. 클래스 내부에 메소드를 만들 때는 다음과 같이 사용합니다.

"""
class 클래스 이름:
    def 메소드 이름(self, 추가적인 매개변수):
        pass
"""

파이썬에서는 메소드를 멤버함수 혹은 인스턴스 함수라 부릅니다.

8-2 클래스의 추가적인 구문

어떤 클래스의 인스턴스인지 확인하기

일단 객체가 어떤 클래스로부터 만들어졌는지 확인할 수 있도록 isinstance()함수를 제공합니다.

“”

isinstance(인스턴스, 클래스)
"""
결과는 불로써 출력됩니다.
“”

```
class Student:
    def __init__(self):
        pass

student = Student()
print(isinstance(student, Student))
>>True
```

이 함수는 상속 관계까지 확인합니다.

Study Title _____ Date _____