

2025-04-24(목)

- 클래스

- [How to apply color on text in Markdown](#)

- 클래스

- 필드와 메서드로 구성
- [9. 클래스](#) ♦ [python.org](#)
 - 클래스는 데이터와 기능을 함께 묶는 방법을 제공
- [05-1 클래스](#) ♦ [wikidocs](#), [점프 투 파이썬](#)
 - 클래스(class)란 똑같은 무언가를 계속 만들어 낼 수 있는 설계 도면(과자 틀), 객체(object)란 클래스로 만든 피조물(과자 틀로 찍어 낸 과자)을 뜻한다.
 - 객체와 인스턴스의 차이
 - 클래스로 만든 객체를 '인스턴스'라고도 한다.
 - `a = Cookie()`로 만든 `a`는 객체이다.
 - `a` 객체는 `Cookie`의 인스턴스이다.
 - 인스턴스라는 말은 특정 객체(`a`)가 어떤 클래스(`Cookie`)의 객체인지를 관계 위주로 설명할 때 사용한다.
 - 'a는 인스턴스'보다 'a는 객체'라는 표현이 어울리며 'a는 `Cookie`의 객체'보다 'a는 `Cookie`의 인스턴스'라는 표현이 훨씬 잘 어울린다.
- () 는 메서드를 의미
- 객체의 상호작용
 - 객체들은 서로간에 기증(동작)을 이용하고, 데이터를 주고 받음
 - [\[Python 33\] 객체들 간의 관계-1 : 컴포지션\(composition\)](#)
 - [AI가 알려주는 Python - 스마트한 Python 학습](#)
 - [4.1 클래스와 객체](#)
 - 객체는 클래스의 속성 값을 저장하고, 클래스의 메서드를 호출하여 객체 간의 상호작용을 가능하게 합니다.
- 객체간의 관계(Relationship)
 - 객체지향 시스템은 상호관계를 맺는 여러 클래스에서 생성된 객체들이 기능을 수행함
 - 객체지향 프로그램에서는 객체는 다른 객체와 관계를 맺음
 - 관계의 종류
 - 집합 관계: 클래스들 사이의 전체 또는 부분과 같은 관계를 나타냄
 - 집약(aggregation)관계와 합성(composition)관계가 존재

- 집약관계 - 전체와 부분이 독립적으로 존재(weak relationship)
 - 합성관계 - 전체가 소멸되면 부분도 같이 소멸(strong relationship)
- 연관 관계(Association)
 - 클래스들이 개념상 서로 연결되었음을 나타냄
 - 실선이나 화살표로 표시
 - 한 클래스가 다른 클래스에서 제공하는 기능을 사용하는 경우를 표시함
- 일반화 관계(Generalization)
 - 객체지향 개념에서는 상속 관계라고 함
 - 한 클래스가 다른 클래스를 포함하는 상위 개념일 때, 이를 IS-A관계라고 하며, UML에서는 일반화 관계로 모델링한다.
 - 속이 빈 화살표를 사용해 표시
- 의존 관계(dependency)
 - 연관 관계와 같이 한 클래스가 다른 클래스에서 제공하는 기능을 사용할 때를 나타냄
 - 차이점은 두 클래스의 관계가 한 메서드를 실행하는 동안과 같은, 매우 짧은 시간만 유지된다는 점
 - 점선 화살표를 사용해 표시
- UML v1.1 개념-UML의 기본-관계(Relationships) ★★
 - 관계의 종류
 - Aggregation (집합연관): 전체와 부분간의 관계를 표현
 - 두 클래스가 연관관계를 갖고 있으면서 전체/부분관계를 띄고 있다면 집합연관으로 표기
 - Composition (합성연관): Composition은 Aggregation의 특수한 경우이며, 전체에 해당하는 클래스가 소멸될 때 부분에 해당하는 클래스도 함께 소멸됨
 - Association (연관): 한 사물의 객체가 다른 사물에 대한 객체와 연결되어 있음을 표현
 - 두 클래스가 서로 Association이 있다면 한쪽 객체에서 다른 객체를 참조 할 수 있음을 의미
 - Generalization(일반화:상속): 일반화 된 개념적인 사물과 특수화된 구체적인 사물과의 관계를 표현
 - 자식은 부모의 속성과 행동을 공유
 - 부모가 사용되는 곳에는 항상 자식으로 대체 할 수 있으며 그 반대는 성립하지 않는다.

- Realization(실체화:구현): 인터페이스와 실제 구현 클래스들과의 관계
 - 정의된 Interface와 이를 구현하는 class를 연결할 때 사용
 - Dependency (의존): 한 클래스가 변경 되었을 때 이것을 사용하는 다른 클래스에 영향이 미치는 관계
 - 한 클래스의 메소드가 다른 클래스의 객체를 인자로 받아 그 메소드를 사용
- SECTION 010 UML - 관계(Relationship)
- [자료구조] 객체 관계 (Object Relationship)
- 함께해요 파이썬 전문가 ♦ wikidocs
 - 파이썬 디자인 패턴 ★ ♦ wikidocs
- 시스템 분석 및 설계 with Visual Paradigm
 - 1-2. UML 다이어그램 종류
- [UML 관계] 구성요소 간의 관계 표기법 ★
- 디자인 패턴을 배워보자 2일차 - UML diagram 개론과 class relationship
- UML 클래스 관계
- [소프트웨어] UML관계, UML다이어그램
- [Python] 파이썬 UML Diagram 자동 생성
- 소프트웨어는 키워드로 암기
- 객체 지향 프로그래밍의 특징
 - 캡슐화(Encapsulation, 정보 은닉)
 - 객체의 필드, 메서드를 하나로 묶고, 실제 구현 내용을 감추는 것
 - 외부 객체는 객체 내부 구조를 알지 못하며, 객체가 노출해 제공하는 필드와 메서드만 이용 가능
 - 필드와 메서드를 캡슐화하여 보호하는 이유
 - 외부의 잘못된 사용으로 인해 객체가 손상되지 않도록 하기 위함
 - 파이썬에서는 명시적인 접근제한자(public, private, protected)키워드가 없음
 - python은 _variable_name(protected), __variable_name(private) 표기로 접근제한자 변수 정의
 - __ : name mangling - 클래스 정의 시점에 변수나 메서드 이름을 내부적으로 변경하는 메커니즘
 - 변수이름이 __name 일 경우, 해당 이름을 _클래스명__name으로 자동 변경함.
 - 주로 subclass에서 우연히 같은 이름을 덮어쓰는 것을 방지하기 위해 사용

- 이름 충돌 방지
- [파이썬] 네임 맨글링 (name mangling) ★
 - 파이썬에서 클래스의 속성 이름을 바꾸는 기능
 - 속성 이름을 _ 두 개로 시작하는 것으로 정의하면, 이 속성은 자동으로 네임 맨글링이 되어 _클래스명_속성이름으로 변경 됨
 - 정보 은닉은 객체 내부의 속성이나 구현 내용을 외부에 노출하지 않는 것을 의미
- [파이썬] 클래스 private와 protected 메소드 사용방법 및 특징 ★ ★
 - Private 메서드
 - Private 메서드는 클래스 내부에서만 사용되도록 의도된 메서드
 - 파이썬에서는 메서드 이름 앞에 두 개의 밑줄(_)을 붙여서 private 메서드를 나타냄
 - Private 메소드는 클래스 내부에서만 직접 호출할 수 있습니다. 이 메소드는 클래스 외부나 서브클래스에서 직접 접근할 수 있음
 - __my_private_method
 - Protected 메서드
 - Protected 메서드는 해당 클래스 및 그 서브클래스에서만 사용되어야 함
 - 메소드 이름 앞에 하나의 밑줄(_)을 붙여서 protected 메서드를 나타냄
 - Protected 메서드는 클래스 내부와 상속받은 서브클래스에서 접근할 수 있음
 - 다만 이는 그렇게 사용하겠다는 약속일 뿐이고 강제성은 없다.
 - _my_protected_method
- [Python] 클래스의 인스턴스 변수 protected, private(접근 제어자)
- Access Modifiers in Python : Public, Private and Protected ★
- 객체지향 (public, private, protected) ★
 - private
 - private로 선언된 attribute, method는 해당 클래스에서만 접근 가능
 - protected
 - protected로 선언된 attribute, method는 해당 클래스 또는 해당 클래스를 상속받은 클래스에서만 접근 가능

- public
 - public으로 선언된 attribute, method는 어떤 클래스라도 접근 가능

[python 접근 제한자 예제]

```
class Parent:
    def __init__(self):
        self.__private_var = "부모 값"
        self._protected_var = "부모 값"
        self.public_var = "부모 값"

    def show_private_value(self):
        print(f"__private_var: {self.__private_var}")

    def show_protected_value(self):
        print(f"_protected_var: {self._protected_var}")

    def show_public_value(self):
        print(f"public_var: {self.public_var}")

class Child(Parent):
    def __init__(self):
        super().__init__()
        self.__private_var = "자식 값"
        self._protected_var = "자식 값"
        self.public_var = "자식 값"

if __name__ == "__main__":
    c = Child()
    c.show_private_value()
    c.show_protected_value()
    c.show_public_value()
    print(f"c.__dict__: {c.__dict__}")
```

[output]

```
__private_var: 부모 값
_protected_var: 자식 값
public_var: 자식 값
c.__dict__: {'_Parent__private_var': '부모 값', '_prot
```

○ 객체 지향 프로그래밍의 특징

- 상속(Inheritance)
 - 상위 객체의 필드와 메소드를 하위 객체에게 물려주는 행위
 - 하위 객체는 상위 객체를 확장해서 추가적인 필드와 메서드를 가질 수 있음.
 - 상속의 효과

- 유지보수의 편리성 제공
- 객체의 다형성 구현
- 다형성(Polymorphism)
 - 같은 타입이지만 실행 결과가 다양한 객체를 대입할 수 있는 성질
 - 부모 타입에는 모든 자식 객체가 대입 가능
 - 인터페이스 타입에는 모든 구현 객체가 대입 가능
 - 효과
 - 객체를 부품화 시키는 것이 가능해짐
 - 유지보수가 용이함 (기능 확장 및 변경이 용이)
- `__main__` : 현재 실행 중인 최상위(메인) 스크립트의 이름임
- pass 키워드
 - 아무 작업도 하지 않고 넘어가는 명령어
 - 파이썬 명령문 (statement)
 - 명령문(statement)이란, 특정 작업을 수행할 수 있는 하나의 코드를 말한다. 다시 말해, 각각의 명령문은 자신에게 주어진 고유의 역할을 가진다.
 - Python의 명령문 ★
 - Python pass Statement
 - The pass statement is used as a placeholder for future code.
 - pass, continue 그리고 break에 대한 이해(python)
 - How to Use the Python pass Statement

[ChaptGPT 답변]

[질문]

파이썬 pass 가 명령어 인가요?

[ChaptGPT 답변]

아주 좋은 질문이에요! 😊

이번엔 pass에 대해 살펴볼게요.

✅ pass는 명령어(command) 일까요?

정확히 말하면,

pass는 파이썬의 "문장(statement)" 중 하나이고,

"아무것도 하지 말라" 는 특별한 지시를 내리는 **키워드(keyword)

🔍 pass는 왜 필요할까요?

pass는 문법적으로 문장이 꼭 필요한 자리인데, 아직 코드를 안

예:

[python 코드]

```
def my_function():
    pass # 나중에 구현할 예정

if True:
    pass # 조건은 맞지만 지금은 실행할 코드 없음
```

🌸 핵심 요약

항목	설명
pass---는 명령어인가요?	❌ 보통의
무엇인가요?	✅ 파이썬의 내장 키워드이며, **문장(statement)**입니다.
역할은요?	➡ "여기 아무 것도 안 할 거야" 라고 명시하는 용도입니다.

컴퓨팅 사고의 정의

- 컴퓨팅 사고란?
 - 인간이 문제를 분석하고 컴퓨터로 문제를 해결하기 위한 절차를 만드는 것
- 니넷 윈의 컴퓨팅 사고(CMU)의 창시자 (2006년 3월 ACM)
 - 컴퓨팅 사고(computational Thinking)는?
 - 누구나 배워서 활용할 수 있는 보편적인 사고 기술
 - 누구나 갖추어야 할 기본적인 역량
 - 모든 아동이 기본적으로 갖춰야 하는 분석 역량에 추가해야 함
 - 문제를 수립하고 해결책을 만들어 ...
- Google의 컴퓨팅 사고 모델

문제해결 단계	컴퓨팅 사고 핵심 개념
문제분석	자료분석(Data Analysis)
자료수집 및 표현	자료수집(Data Collection)
자료수집 및 표현	자료표현(Data Representation)
분해	분해(Decomposition)
패턴인식	패턴인식(Pattern Recognition)
추상화	추상화(Abstraction)
알고리즘	알고리즘(Algorithm)
시뮬레이션	시뮬레이션(Simulation)

병렬화	병렬화(Parallelization)
자동화	자동화(Automation)

- 문제 해결과 컴퓨팅 사고 개요
 - 문제 분석 및 표현(Analysis & Representation)
 - 논리적으로 사고하는 것
 - 분해(Decomposition)
 - 복잡한 문제를 좀 더 작고 처리가 가능한 부분으로 분해
 - 패턴인식
 - 추상화(Abstraction)
 - 관련 없는 세부 사항은 무시하고 중요한 정보에만 집중
 - 알고리즘(Algorithms)
 - 문제에 대한 **단계적인 해결책**을 개발
 - 컴퓨팅 사고
 - 추상화 및 패턴 인식 사용 - 새롭고 다양한 방식으로 문제 표현
 - 데이터를 논리적으로 구성, 분석
 - **문제를 작은 부분 문제들로 나누어서 해결**
 - 컴퓨팅 사고가 아닌 것
 - 소프트웨어 사용 학습 X
 - 컴퓨터처럼 생각 XXX
 - 컴퓨팅 사고(Computational Thinking)의 정의
 - 문제를 수립하고 해결책을 만들어 컴퓨팅 시스템을 통해 효과적으로 수행되도록 표현하게 하는 사고 과정
 - 컴퓨팅 사고 모델(Computatioin Thinking Model)
 - 문제 분석, 자료 수집 및 표현, 분해, 패턴인식, 추상화, 알고리즘, 시뮬레이션, 병렬화, 자동화