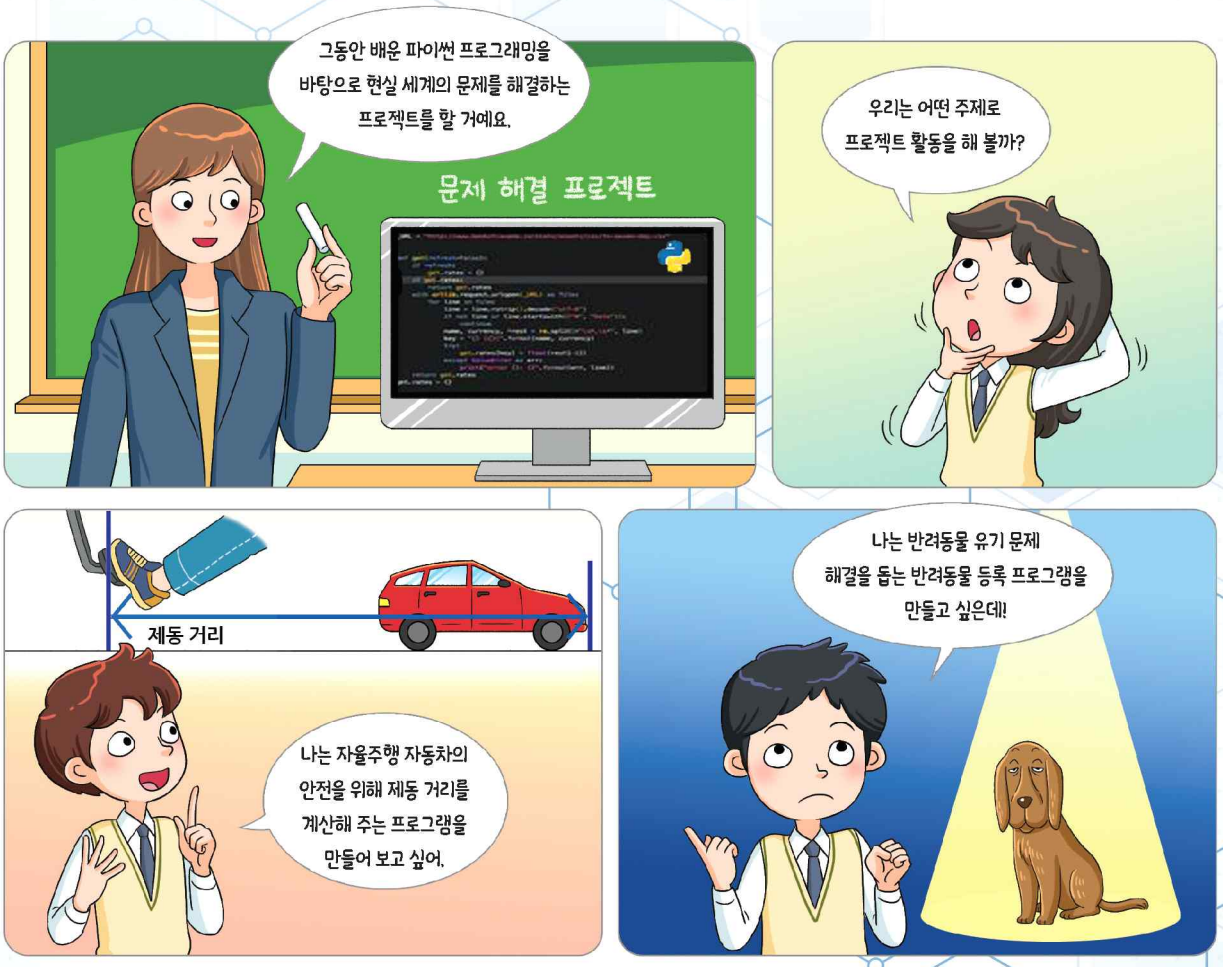
08 협력적 문제 해결 프로젝트

✚ **학습 목표** •문제 해결을 위한 프로그램을 협력적으로 설계하고 구현할 수 있다.
•문제 해결을 위한 프로그램의 성능을 평가하고 공유할 수 있다.

✚ **학습 요소** •프로그래밍 프로젝트, 성능 평가

생각 깨우기 문제 해결 프로젝트 주제를 생각해 보자.



Q 프로그래밍으로 해결할 수 있는 주제는 어떤 것이 있을까?

1 | 프로그래밍 프로젝트란?

실생활에서 프로그램은 우리의 삶을 편리하게 한다. 일정을 관리하는 일정 관리 프로그램, 수입, 지출을 정리하는 가계부 프로그램, 다른 사람과의 소통을 돕는 메신저 프로그램, 재미있고 유익한 영상 시청을 돕는 영상 시청 프로그램 등을 매일 사용할 정도로 우리의 삶과 프로그램은 밀접한 관계가 있다.

지금까지 학습한 내용을 바탕으로 프로그래밍을 통해 나와 우리 주변의 실생활 문제나 다양한 학문 분야의 문제를 해결해 보려고 한다. 이와 같은 문제 해결 프로그래밍은 혼자 수행하는 것보다 여러 사람의 의견을 듣고, 함께 협력하는 협력 프로젝트 형태로 수행하는 것이 효율적이다.

문제 해결 프로젝트는 계획, 구현, 개선하는 과정을 포함하며, 프로젝트의 절차는 주제 선정 및 문제 정의, 추상화, 알고리즘 설계, 프로그램 작성, 성능 평가 및 공유 단계로 이루어진다.

문제를 정의하는 과정에서 우리와 관련 있는 문제인지, 많은 사람이 필요로 하는 문제인지, 프로그램으로 해결할 수 있는 문제인지 생각해 본다.

브레인스토밍

자유로운 분위기 속에서 다양한 아이디어를 제안하며 가장 좋은 아이디어를 선정하거나 아이디어를 조합하여 새로운 아이디어를 창조해 내는 방법이다.

1단계 주제 선정 및 문제 정의	실생활 혹은 여러 학문 분야의 주제를 생각해 보거나 브레인스토밍을 통해 다양한 주제를 탐색하여 주제를 선정한다. 이를 바탕으로 해결하고자 하는 문제를 구체적으로 정의한다.
2단계 추상화	문제의 복잡성을 제거하고 해결하기 쉬운 형태로 표현하는 과정이다. 추상화는 문제 분석, 문제 분해, 모델링 과정으로 구성된다. **문제 분석** 문제를 분석하여 초기 상태와 목표 상태를 설정하고, 수행 작업을 정의한다. **문제 분해** 복잡한 문제를 해결 가능한 작은 문제로 나눈다. **모델링** 복잡한 문제를 쉽게 이해하여 해결할 수 있는 형태로 다시 표현한다. 이를 위해 글, 그림, 그래프, 표 등을 사용하여 구조화한다.
3단계 알고리즘 설계	분해한 핵심 요소를 참고하여 작은 문제별로 문제 해결을 위한 알고리즘을 설계한다. 알고리즘은 자연어, 순서도, 의사 코드 중 편리한 방법으로 작성한다.
4단계 프로그램 작성	설계한 알고리즘에 따라 프로그램을 작성한다.
5단계 성능 평가 및 공유	주어진 입력에 대해 의도한 대로 자료를 처리해 결과를 출력하는지 확인한다. 또한 분해한 작은 문제를 모두 해결하여 처음 설정한 큰 문제를 효율적으로 해결했는지 확인한다. 정의한 문제를 해결할 수 있다면 프로그램을 공유하고, 피드백을 반영하여 최종 완성한다.

알고 가기 디버깅(Debugging)

그레이스 호퍼(Grace Hopper)는 초기 컴퓨터 개발자로, 컴퓨터 고장의 원인을 찾던 중 회로에 나방 한 마리가 끼여 있는 것을 발견했다. 이때 벌레(Bug)를 제거한다는 의미가 잘못된 부분을 고친다는 의미로 활용되며, 디버깅이라는 용어가 생겼다.

디버깅은 프로그램의 실행 결과가 원하는 대로 나오지 않거나 오류가 발생했을 때 프로그램의 코드를 처음부터 한 줄씩 확인하며 오류가 있는 부분을 찾아 수정하는 과정을 뜻한다.

프로젝트 1 안전한 자율주행 자동차 만들기

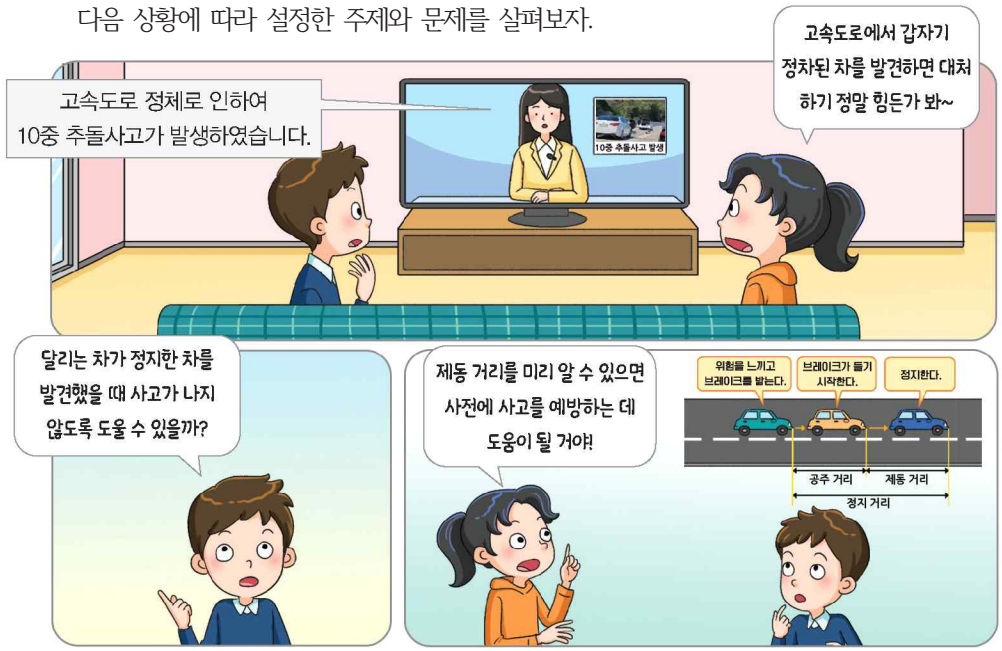
✿ 프로그래밍 프로젝트 진행을 위해 모둠을 구성하고, 모둠원 간 역할을 나누어 보자.

예 모둠장, 프로그래머, 발표자, 기획자 등 자유롭게 선정

모둠명:	
이름	역할

1단계 주제 선정 및 문제 정의

다음 상황에 따라 설정한 주제와 문제를 살펴보자.



제동 거리를 계산하는 프로그램을 만들어 보세요!

주제 선정 과학 문제 해결

문제 정의 제동 거리 계산 프로그램 만들기

2단계 추상화

'제동 거리 계산 프로그램 만들기'에 대해 문제 분석, 문제 분해, 모델링 등의 추상화 단계를 수행해 보자.

문제 분석

문제의 초기 상태와 목표 상태를 분석하여 정의한 문제를 해결하기 위한 목표를 명확하게 설정한다.

초기 상태 제동 거리 계산이 되지 않은 상태

목표 상태 제동 거리를 계산한 상태

✿ 제동 가속도는 브레이크를 밟았을 때의 가속도를 의미하며, 속도가 줄어드는 감속의 의미를 갖는다.

문제 분해

복잡한 문제를 작은 문제로 분해한다.

문제 제동 거리 계산 프로그램 만들기

작은 문제
- 자동차의 현재 속도(km/h)와 제동 가속도(m/s^2)의 크기 입력받기
- 제동 거리(m) 계산하기
- 결과 출력하기

모델링

작은 문제를 바탕으로 글, 그림, 그래프, 표 등을 사용하여 구조화한다.

현재 속도와 제동 가속도의 크기 입력을 기다린다. → 자동차의 현재 속도와 제동 가속도의 크기를 입력받는다. → 제동 거리를 계산하고 출력한다.

3단계 알고리즘 설계

작은 문제별로 문제 해결을 위한 알고리즘을 설계해 보자.,

작은 문제	알고리즘
자동차의 현재 속도와 제동 가속도의 크기 입력받기	❶ '자동차의 현재 속도를 입력하세요.' 출력하기 ❷ 자동차의 현재 속도를 입력받아 변수 v1에 실수로 저장하기 ❸ '자동차의 제동 가속도의 크기를 입력하세요.' 출력하기 ❹ 자동차의 제동 가속도의 크기 입력받아 변수 a에 실수로 저장하기
제동 거리 계산하기	❶ 현재 속도(v1)의 단위를 변경하여 변환된 현재 속도(v2)에 저장한다. ❷ 제동 거리 변수(result)에 제동 거리 계산 결과를 저장한다.
결과 출력하기	❶ '제동 거리는 다음과 같습니다.' 출력하기 ❷ 제동 거리 출력하기

입력받은 현재 속도의 단위는 km/h이므로 가속도의 단위와 거리, 시간의 단위를 통일하기 위하여 현재 속도(v1)*1000/3600으로 단위를 통일한다.

Tip 움직이는 물체의 제동 거리를 계산하는 방법

$$\text{제동 거리} = \frac{\text{현재 속도}^2}{(2*\text{제동 가속도 크기})}$$

4단계 프로그램 작성

설계한 알고리즘을 바탕으로 프로그램을 작성해 보자.

프로그램

```
1  # 자동차의 현재 속도(v1)와 제동 가속도의 크기(a) 입력받기
2  v1 = float(input("자동차의 현재 속도를 입력하세요:"))
3  a = float(input("자동차의 제동 가속도의 크기를 입력하세요:"))
4
5  # 현재 속도의 단위 변경하기
6  v2 = v1 * 1000 / 3600
7
8  # 제동 거리 계산하기
9  result = v2 * v2 / (2 * a)
10
11 # 결과 출력하기
12 print('제동 거리는 다음과 같습니다:', result, 'm')
```

실행 결과

입력

```
자동차의 현재 속도를 입력하세요:100
자동차의 제동 가속도의 크기를 입력하세요:15
```

출력

```
제동 거리는 다음과 같습니다:25.720164609053498 m
```

설명 자동차의 현재 속도가 100km/h일 때 제동 가속도의 크기가 3000km/h^{2}이라면 제동 거리는 약 0.128km임을 알 수 있다.

5단계 성능 평가 및 공유

목표 상태를 달성하기 위한 프로그램이 올바르게 수행되는지 확인하기 위해 성능 평가 항목을 작성하고 결과를 확인해 보자.

성능 평가 항목	결과
자동차의 현재 속도와 제동 가속도의 크기를 잘 입력받았는가?	○ | ✕
현재 속도의 단위가 잘 변경되는가?	○ | ✕
제동 거리가 정확히 계산되는가?	○ | ✕
제동 거리가 잘 출력되는가?	○ | ✕

해 보기 1 프로그램 개선하기

다음을 참고하여 자동차의 정지 거리를 계산하는 프로그램으로 개선해 보자.

- 자동차의 정지 거리는 제동 거리와 공주 거리의 합으로 구한다.
- 공주 거리는 현재 속도×공주 시간으로 구할 수 있다. (단, 속도의 단위는 km/h, 시간의 단위는 시(hour)로 설정한다.)

※ 공주 시간이란 운전자가 전방의 위험 상황을 발견하고 실제 브레이크를 밟아 제동이 걸리기 시작할 때까지의 시간을 의미한다.
※ 일반적으로 공주 시간은 초 단위로 알 수 있다. 따라서 '공주 시간(초) / 3600'을 하면 시(hour) 단위 공주 시간을 구할 수 있다.

1 공주 시간(s)을 입력받은 후 공주 거리(result2)를 계산하고 출력하는 코드를 작성해 보자.

프로그램

2 정지 거리(result3)를 계산하고 출력하는 코드를 작성해 보자.

프로그램

프로젝트 2 반려동물에 대한 책임감 갖기

1단계 주제 선정 및 문제 정의

다음 상황에 따라 설정한 주제와 문제를 살펴보자.

주제 선정 사회 문제 해결

문제 정의 반려동물 등록 프로그램 만들기

2단계 추상화

'반려동물 등록 프로그램 만들기' 문제에 대해 문제 분석, 문제 분해, 모델링 등의 추상화 단계를 살펴보자.

문제 분석

문제의 초기 상태와 목표 상태를 분석하여 정의한 문제를 해결하기 위한 목표를 명확하게 설정한다.

초기 상태 반려동물 등록이 되지 않은 상태

목표 상태 반려동물 등록이 된 상태

문제 분해

복잡한 문제를 작은 문제로 분해한다.

문제 반려동물 등록 프로그램 만들기

작은 문제
- 반려동물 등록하기(명령 1)
- 반려동물 삭제하기(명령 2)
- 반려동물 목록 조회하기(명령 3)
- 프로그램 종료하기(명령 4)

모델링

작은 문제를 바탕으로 글, 그림, 그래프, 표 등을 사용하여 구조화한다.

반려동물 등록 프로그램의 명령(1~4)을 기다린다. → 반려동물을 등록 혹은 삭제한다. → 반려동물 목록을 조회하거나 프로그램을 종료한다.

3단계 알고리즘 설계

작은 문제별로 문제 해결을 위한 알고리즘을 살펴보자.

작은 문제	알고리즘
반려동물 등록하기	❶ 반려동물을 등록하기 위한 클래스를 생성한다. • 속성: 등록 번호(Number), 동물 이름(A_name), 동물 종류(Species), 등록자(R_name) • 메서드: 객체의 모든 속성 출력 ❷ 빈 리스트를 생성한다. ❸ 사용자의 명령 1~4 중 하나를 무한히 반복하며 입력받는다. ❹ 사용자가 1(등록)을 입력한다. ❺ 사용자가 입력한 반려동물의 정보(등록 번호(Number), 동물 이름(A_name), 동물 종류(Species), 등록자(R_name)) 객체를 생성하고, 리스트에 추가한다. ❻ '등록되었습니다.'를 출력한다.
반려동물 삭제하기	❶ 사용자가 2(삭제)를 입력한다. ❷ 삭제하고자 하는 반려동물의 이름과 등록자의 이름을 입력한다. ❸ 리스트에서 해당 반려동물을 삭제한다. ❹ '삭제되었습니다.'를 출력한다.
반려동물 목록 조회하기	❶ 사용자가 3(조회)을 입력한다. ❷ 반려동물 목록을 출력한다.
프로그램 종료하기	❶ 사용자가 4(종료)를 입력한다. ❷ 무한 반복을 마치며 프로그램을 종료한다.

4단계 프로그램 작성

설계한 알고리즘을 바탕으로 한 프로그램을 살펴보자.

프로그램

```
# 클래스 정의하기
class Register:
  def __init__(self, Number, A_name, Species, R_name):
    self.Number = Number
    self.A_name = A_name
```

```
        self.Species = Species
        self.R_name = R_name

    def print_object(self):                          # 객체의 모든 속성을 출력
        print('등록 번호: ', self.Number)
        print('동물 이름: ', self.A_name)
        print('동물 종류: ', self.Species)
        print('등록자: ', self.R_name)
        print('')

# 등록, 출력, 삭제, 주문 함수 정의하기
def input_register():                                # 등록하기
    Number = input('등록 번호를 입력해 주세요. ')
    A_name = input('동물 이름을 입력해 주세요. ')
    Species = input('동물 종류를 입력해 주세요. ')
    R_name = input('등록자를 입력해 주세요. ')
    register = Register(Number, A_name, Species, R_name)
    print('등록되었습니다.')
    return register

def print_register(register_list):                   # 등록된 모든 동물을 출력
    for register in register_list:
        register.print_object()

def delete_register(register_list, A_name, R_name): # 등록된 모든 동물을 삭제
    for i, register in enumerate(register_list):
        if register.A_name == A_name and register.R_name == R_name:
            del register_list[i]
            print('삭제되었습니다.')

def print_order():                                   # 명령을 입력받는 함수
    print('1. 등록')
    print('2. 삭제')
    print('3. 조회')
    print('4. 종료')
    order = int(input('숫자를 입력해 주세요.'))
    return order

# 메인 프로그램 실행
register_list = []
while 1:
    order = print_order()
    if order == 1:                                   # 등록
        register = input_register()
        register_list.append(register)
    elif order == 2:                                 # 삭제
        A_name = input('삭제할 동물 이름을 입력해 주세요: ')
        R_name = input('등록자를 입력해 주세요: ')
        delete_register(register_list, A_name, R_name)
    elif order == 3:                                 # 조회
        print_register(register_list)
    elif order == 4:                                 # 종료
        print('프로그램을 종료합니다.')
        break
```

enumerate() 함수란?

파이썬의 내장 함수로 리스트의 인덱스와 원소에 동시에 접근하기 위해 활용하는 함수다.

```
for i, j in enumerate(['A','B','C']):
    print(i, j)
```

위 코드를 실행했을 때 출력 결과는 아래와 같다.

```
0 A
1 B
2 C
```

5단계 성능 평가 및 공유

프로그램 실행해 보기

❶ 반려동물을 3마리 이상 등록해 보자.
❷ 입력한 반려동물의 목록을 조회해 보자.
❸ 첫 번째로 등록한 반려동물을 삭제하고, 삭제된 목록을 조회해 보자.
❹ 프로그램을 종료해 보자.

프로그램이 효율적으로 작성되었는지 확인하고, 아래 표의 평가 항목을 작성해 보자.

성능 평가 항목	결과
반려동물 등록하기가 잘 실행되는가?	○ | ✕
반려동물 삭제하기가 잘 실행되는가?	○ | ✕
반려동물 목록 조회하기가 잘 실행되는가?	○ | ✕
프로그램 종료하기가 잘 실행되는가?	○ | ✕

해 보기 2 프로그램 개선점 작성하기

'반려동물 등록 프로그램'을 개선하기 위한 아이디어를 함께 생각해 보자.

소단원 1 요약

❶ 문제 해결 프로그래밍 프로젝트는 계획, 구현, 개선하는 과정을 포함한다.

❷ 프로젝트 절차: 주제 선정 및 문제 정의 → 추상화 → 알고리즘 설계 → 프로그램 작성 → 성능 평가 및 공유

소단원 자기 평가

평가 항목	평가 기준		
	잘함	보통	노력
1. [과정·기능] 실생활 및 다양한 학문 분야에서의 문제 해결을 위한 알고리즘을 설계할 수 있다.			
2. [과정·기능] 실생활 및 다양한 학문 분야에서의 문제 해결을 위한 프로그램을 구현할 수 있다.			
3. [가치·태도] 문제 해결을 위해 설계한 알고리즘을 구현하려는 실천적 자세를 인식할 수 있다.			