Q1. 중간고사 기말고사 점수를 따로 받아 저장하는 클래스를 구현해보세요. 단, 생성자의 인스턴스는 private으로 선언되어야하며, 데코레이터를 이용해 데이터를 저장하고, 함수를 이용해 평균값을 출력해보세요.

〈문제 풀이 핵심〉 : 생성자의 인스턴스를 1.Private Variable 으로 선언해야 하며 2.Decorator을 이용해 데이터를 저장하고 함수를 이용해 평균값을 계산하는 문제입니다.

1. Private Variable (프라이빗 변수)란?

파이썬은 내부의 변수를 외부에서 사용하는 것을 막고 싶을 때 인스턴스 변수 이름을 __〈변수이름〉 형태로 선언합니다. 프라이빗 변수를 선언함으로서 해당 변수를 외부에서 접근할 수 없습니다.

2. Decorator (@property)란?

프라이빗 변수는 기본적으로 외부에서 접근할 수 없습니다, 하지만 내부의 메소드를 통해서 접근할 수 있도록 구현할 수 있습니다. 그 방법은, 해당 메소드 앞에 @property 모양의 데코레이터를 선언하는 것입니다.

프라이빗 변수에 접근할 수 있는 데코레이터는 크게 두가지가 있습니다. 변수를 불러오는(또는 추출하는) 함수(getter)와 변경하는 함수(setter)입니다. getter를 선언하는 방법은 강의에서 했던것 처럼 앞에 @property를 작성하는 것입니다. setter를 선언하는 방법은 @(함수명).setter를 작성하는 것입니다.

3. 풀이 및 소스 코드 :

풀이 1:

```
In [40]: # mid - 50 / final - 75

class Score():
    def __init__(self, mid, final):
        self.__mid = mid
        self.__final = final
        print(f"중간고사 점수 : {mid}")
        print(f"기말고사 점수 : {final}")

    @property
    def mid(self):
        return self.__mid

    @mid.setter
    def mid(self, mid):
        self.__mid = mid
        print(f"중간고사 점수 : {mid}")

    @property
    def final(self):
        return self.__final

    @final.setter
    def final(self, final):
        self.__final = final

    def average(self):
        average = (self.__mid + self.__final) / 2
        return average
```

```
In [43]: score = Score(50, 75)
print('평균 점수 :', score.average())
```

```
중간고사 점수 : 50
기말고사 점수 : 75
평균 점수 : 62.5
```

- 생성자의 인스턴스는 private으로 선언했습니다 : self.__mid / self.__final
- 데코레이터를 이용해 데이터를 저장하고 접근합니다 : @property / @mid.setter , @property / @final.setter, 데코레이터를 선언해줌으로서 사용자의 의도에 맞게 자동적으로 값을 추출하거나 수정해주는 기능을 합니다.

풀이2:

```
# 풀이 1
class Score():
    def __init__(self, mid, final):
        self.__mid = mid
        self.__final = final

    def score_mid(self):
        return print(self.__mid)

    def score_final(self):
        return print(self.__final)

    def score_average(self):
        return ((self.__mid + self.__final)/2)
```

```
#출력함수
score = Score(50, 75)
score.score_mid()
score.score_final()
score.score_average()
```

```
50
75

62.5
```

- 생성자의 인스턴스를 private로 선언했을시에, 외부에서는 해당 변수를 접근할 수 없지만, 내부적으로는 접근할 수 있다라는 특징을 이용했습니다. getter와 setter 데코레이터를 사용하지 않고 위와 같이 간단하게 작성할 수도 있습니다.
- 도중에 점수를 변경할 수 없다는 단점이 있습니다.

Q2. 다양한 탈 것을 사용하는 게임을 만드는 중입니다. 빠른 구현을 위해서 이미 구현한 Car 클래스를 이용해서 Bike라는 클래스를 새로 제작하려고 합니다. Car 클래스를 상속받아서 새로운 Bike 클래스를 아래와 같이 출력되도록 구성해보세요.

〈문제 풀이 핵심〉 : 새로운 클래스를 만들어서 기존의 클래스를 상속받아 확장하는 문제입니다.

1. 상속이란?

상속이란, 새로운 클래스(자식 클래스)의 파라미터에 기존 클래스(부모 클래스)를 넣어줌으로서, 자식 클래스는 부모 클래스의 모든 속성 정보를 갖고 있는 것입니다.

2. 풀이 및 소스 코드 :

풀이 1:

In [3]:
```
class Car():
    def __init__(self, fuel, wheels):
        self.fuel = fuel
        self.wheels = wheels

class Bike(Car):
    def __init__(self, fuel: str, wheels: int, size: str):
        super().__init__(fuel, wheels)
        self.size = size
```

In [4]:
```
bike = Bike("gas", 2, "small")
print(bike.fuel, bike.wheels, bike.size)
```

```
gas 2 small
```

- Bike는 파라미터로 Car클래스를 받아 이를 상속받습니다.
- 자식 클래스 Bike는 부모 클래스 Car의 생성자에 size라는 속성을 새롭게 추가해서 확장하고 있습니다.
- 팀원이 풀이를 할때, wheels와 size의 데이터타입을 혼동하는 경우가 있었습니다. 이를 방지하기 위해, 1주차에 배웠던 데이터 타입 hint를 추가했습니다.

Q3. 이번 시험 결과에 대한 데이터를 학과 사무실에서 CSV파일로 전달해줬습니다. 우리는 이 파일을 이용해서 데이터 처리를 진행해야 합니다. 파일 입출력을 이용해 파일 데이터를 리스트로 만들어보세요.

〈문제 풀이 핵심〉 : csv파일 읽어와서 데이터를 리스트 형태로 변형해 출력하는 문제입니다.

1. csv파일 읽기 (csv module)

파일의 경로를 변수로 저장합니다. read_file이라는 함수는 해당 경로의 파일을 csv모듈을 이용해서 읽어옵니다. (csv모듈은 csv형식의 표 형식 데이터를 읽고 쓰는 기능을 도와줍니다.).

csv파일을 csv모듈을 이용해 저장한 변수는, 지정된 *csvfile*의 줄을 이터레이트 하는 판독기(reader) 객체입니다. 반복문을 통해서 데이터를 리스트에 저장합니다. 해당 파일에서 원하는 작업을 완료했으면 close를 합니다.

2. 풀이 및 소스 코드 :

풀이 1:

```
In [19]: import csv
         import pprint

         class ReadCSV():
             def __init__(self, file_path):
                 self.file_path = file_path

             def read_file(self):
                 data_list = list()
                 score_file = open(self.file_path, "r" )
                 csv_data = csv.reader(score_file)

                 for idx in csv_data:
                     data_list.append(idx)

                 score_file.close()

                 return data_list
```

```
In [20]: file_path = "./csv_files/data-01-test-score.csv"
         read_csv = ReadCSV(file_path)
         pprint.pprint(read_csv.read_file())
```

```
[['73', '80', '75', '152'],
 ['93', '88', '93', '185'],
 ['89', '91', '90', '180'],
 ['96', '98', '100', '196'],
 ['73', '66', '70', '142'],
 ['53', '46', '55', '101'],
 ['69', '74', '77', '149'],
 ['47', '56', '60', '115'],
 ['87', '79', '90', '175'],
 ['79', '70', '88', '164'],
 ['69', '70', '73', '141'],
 ['70', '65', '74', '141'],
 ['93', '95', '91', '184'],
 ['79', '80', '73', '152'],
 ['70', '73', '78', '148'],
 ['93', '89', '96', '192'],
 ['78', '75', '68', '147'],
 ['81', '90', '93', '183'],
 ['88', '92', '86', '177'],
 ['78', '83', '77', '159'],
 ['82', '86', '90', '177']
```

- csv모듈을 이용해 저장한 csv_data객체를 for문을 이용해서 data_list에 저장합니다.
- data_list를 완성했으면, csv파일을 close하고 data_list를 반환합니다.
- csv모듈을 이용해 데이터를 저장하게 되면 위와 같이 string 데이터 타입으로 저장되는 한계가 있습니다.

풀이 2:

In [1]:
```
import csv
from pprint import pprint

class ReadCSV():

    def __init__(self, file_path :str):
        self.file_path = file_path
        self.lst = self.read_file()

    def read_file(self):
        data_set = []
        with open(self.file_path, "r", encoding="cp949") as p_file:
            csv_data = csv.reader(p_file)
            for row in csv_data:
                data_set.append(list(map(int, row)))
        return data_set
```

In [3]:
```
file_path = "./csv_files/data-01-test-score.csv"
read_csv = ReadCSV(file_path)
pprint(read_csv.read_file())
```

```
[[73, 80, 75, 152],
 [93, 88, 93, 185],
 [89, 91, 90, 180],
 [96, 98, 100, 196],
 [73, 66, 70, 142],
 [53, 46, 55, 101],
 [69, 74, 77, 149],
 [47, 56, 60, 115],
 [87, 79, 90, 175],
 [79, 70, 88, 164],
 [69, 70, 73, 141],
 [70, 65, 74, 141],
 [93, 95, 91, 184],
 [79, 80, 73, 152],
 [70, 73, 78, 148],
 [93, 89, 96, 192],
 [78, 75, 68, 147],
 [81, 90, 93, 183],
 [88, 92, 86, 177],
 [78, 83, 77, 159],
 [82, 86, 90, 177],
 [86, 82, 89, 175],
 [78, 83, 85, 175],
 [76, 83, 71, 149],
 [96, 93, 95, 192]]
```

- 풀이1의 한계를 극복하기 위해서 데이터를 string 데이터 타입이 아닌, int 데이터 타입으로 변환해 저장하는 방법입니다.
- csv모듈로 생성한 csv_data파일을 map함수로 데이터들을 각각 int타입으로 mapping시켜준 후에 리스트 형태로 저장합니다 : data_set.append(list(map(int, row)))
- 이 풀이는 'with open ~' 함수를 활용함으로서 close를 별도로 하지 않아도 되는 장점이 있습니다.

Q4. 우리는 방금 CSV 파일을 읽는 함수를 구현해보았습니다. 하지만 이를 조금 더 효율적으로 사용하기 위해서 클래스로 구성을 진행하려고 합니다. 방금 구현한 함수를 포함한 클래스를 구현해보세요.

<문제 풀이 핵심> : 리스트 내의 자료형을 변환하고, 변환한 데이터들의 합을 출력하는 문제입니다.

현재 read_file()을 통해 data_list에 저장되어 있는 데이터는 전부 string타입입니다. 각 행(row)에 있는 숫자들의 합을 반환하기 위해서는, 숫자 계산할 수 있는 데이터타입인 int로 변환을 해야합니다.

1. 리스트 내의 자료형을 변환 / 데이터들의 합 : 이중 반복문 활용

이중 반복문을 활용해서 데이터 타입을 변환함과 동시에 합을 계산해 반환할 수 있습니다. 첫번째 for문은 data_list의 각 행(row)들을 불러옵니다. 두번째 for문은 행 내부에 있는 각 문자열을 불러옵니다. 불러온 문자열을 int로 형 변환을 해준후에 summation이라는 변수에 중첩해서 더해줍니다. 하나의 행이 끝나면, summation값을 merged_list라는 새로운 리스트에 추가해주고, 새로운 행의 합을 구하기 위해 summation 값을 0으로 초기화해줍니다.

2. 풀이 및 소스 코드 :

풀이 1:

In [28]:
```
import csv
import pprint

class ReadCSV():
    def __init__(self, file_path):
        self.file_path = file_path
        self.data_list = list()
        self.merged_list = list()

    def read_file(self):
        data_list = list()
        score_file = open(self.file_path, "r" )
        csv_data = csv.reader(score_file)

        for idx in csv_data:
            data_list.append(idx)
        self.data_list = data_list

        score_file.close()
        return data_list

    def merge_list(self):
        merged_list = list()

        for row in self.data_list:
            summation = 0
            for num in row:
                summation += int(num)
            merged_list.append(summation)

        self.merged_list = merged_list

        return merged_list
```

In [30]:
```
file_path = "./csv_files/data-01-test-score.csv"
read_csv = ReadCSV(file_path)

pprint.pprint(read_csv.read_file())
print(read_csv.merge_list())
```

In [30]:
```
file_path = "./csv_files/data-01-test-score.csv"
read_csv = ReadCSV(file_path)

pprint.pprint(read_csv.read_file())
print(read_csv.merge_list())
```

```
[['73', '80', '75', '152'],
 ['93', '88', '93', '185'],
 ['89', '91', '90', '180'],
 ['96', '98', '100', '196'],
 ['73', '66', '70', '142'],
 ['53', '46', '55', '101'],
 ['69', '74', '77', '149'],
 ['47', '56', '60', '115'],
 ['87', '79', '90', '175'],
 ['79', '70', '88', '164'],
 ['69', '70', '73', '141'],
 ['70', '65', '74', '141'],
 ['93', '95', '91', '184'],
 ['79', '80', '73', '152'],
 ['70', '73', '78', '148'],
 ['93', '89', '96', '192'],
 ['78', '75', '68', '147'],
 ['81', '90', '93', '183'],
 ['88', '92', '86', '177'],
 ['78', '83', '77', '159'],
 ['82', '86', '90', '177'],
 ['86', '82', '89', '175'],
 ['78', '83', '85', '175'],
 ['76', '83', '71', '149'],
 ['96', '93', '95', '192']]
[380, 459, 450, 490, 351, 255, 369, 278, 431, 401, 353, 350, 463, 384, 369, 470, 368, 447, 443, 397, 435, 432
, 421, 379, 476]
```

- 앞서 설명한 이중 반복문을 활용한 데이터 타입 변환과 데이터의 합을 구하는 방법입니다. def merged_list (self) 함수 내부에 이중 for문을 활용했습니다.

풀이 2:

In [5]:
```
import csv
from pprint import pprint

class ReadCSV():

    def __init__(self, file_path :str):
        self.file_path = file_path
        self.lst = self.read_file()

    def read_file(self):
        data_set = []
        with open(self.file_path, "r", encoding="cp949") as p_file:
            csv_data = csv.reader(p_file)
            for row in csv_data:
                data_set.append(list(map(int, row)))
        return data_set

    def merge_list(self):
        if not self.lst:
            return None
        data_set = []
        for row in self.lst:
            data_set.append(sum(row))

        return data_set

filepath = "./csv_files/data-01-test-score.csv"
read_csv = ReadCSV(filepath)
pprint(read_csv.read_file())
print(read_csv.merge_list())
```

```
[[73, 80, 75, 152],
 [93, 88, 93, 185],
 [89, 91, 90, 180],
 [96, 98, 100, 196],
 [73, 66, 70, 142],
 [53, 46, 55, 101],
 [69, 74, 77, 149],
 [47, 56, 60, 115],
 [87, 79, 90, 175],
 [79, 70, 88, 164],
 [69, 70, 73, 141],
 [70, 65, 74, 141],
 [93, 95, 91, 184],
 [79, 80, 73, 152],
 [70, 73, 78, 148],
 [93, 89, 96, 192],
 [78, 75, 68, 147],
 [81, 90, 93, 183],
 [88, 92, 86, 177],
 [78, 83, 77, 159],
 [82, 86, 90, 177],
 [86, 82, 89, 175],
 [78, 83, 85, 175],
 [76, 83, 71, 149],
 [96, 93, 95, 192]]
[380, 459, 450, 490, 351, 255, 369, 278, 431, 401, 353, 350, 463, 384, 369, 470, 368,
447, 443, 397, 435, 432, 421, 379, 476]
```

- 이 풀이는 앞서 3번에서 제시했던 2번째 풀이의 연장선입니다. read_file함수에서 미리 데이터 타입을 mapping을 활용해 변환해서 저장했다면, merged_list 메소드가 매우 간결해지는 장점이 있습니다.
- 데이터 타입을 변환하지 않아도 되므로, 단순히 sum함수로 합을 구해 data_set에 추가해줍니다.
- 만일 파라미터로 들어온 리스트의 데이터가 비어있다면 None을 반환해줍니다 : if not self.lst : return None

Q5. 이전에 구현한 클래스에서 merge_list의 함수 동작을 변경해야합니다. 단순합계가 아닌 평균을 구하는 함수로 변경해보세요.

<문제 풀이 핵심> : 앞서 구했던 각 행들의 합에서 평균을 구하고, 오름차순으로 정렬하는 문제입니다.

1. 평균 계산

앞서 4번에서 구했던 합들을 각 행의 길이(length)로 나누어 평균을 계산합니다.

2. 오름차순 정렬

merged_list에 평균 데이터가 모두 들어가있습니다. 이를 오름차순으로 정렬하기 위해서 .sort()함수를 활용합니다.

3. 풀이 및 소스 코드 :

풀이1:

In [7]:
```
import csv
import pprint

class ReadCSV():
    def __init__(self, file_path):
        self.file_path = file_path
        self.data_list = list()
        self.merged_list = list()

    def read_file(self):
        data_list = list()
        score_file = open(self.file_path, "r" )
        csv_data = csv.reader(score_file)

        for idx in csv_data:
            data_list.append(idx)
        self.data_list = data_list

        score_file.close()
        return data_list

    def merge_list(self):
        merged_list = list()

        for row in self.data_list:
            summation = 0
            length = len(row)
            for num in row:
                summation += int(num)

            average = summation/length
            merged_list.append(average)

        merged_list.sort()
        self.merged_list = merged_list

        return merged_list
```

In [8]:
```
file_path = "./csv_files/data-01-test-score.csv"
read_csv = ReadCSV(file_path)

pprint.pprint(read_csv.read_file())
print(read_csv.merge_list())
```

```
[['73', '80', '75', '152'],
 ['93', '88', '93', '185'],
 ['89', '91', '90', '180'],
 ['96', '98', '100', '196'],
 ['73', '66', '70', '142'],
 ['53', '46', '55', '101'],
 ['69', '74', '77', '149'],
 ['47', '56', '60', '115'],
 ['87', '79', '90', '175'],
 ['79', '70', '88', '164'],
 ['69', '70', '73', '141'],
 ['70', '65', '74', '141'],
 ['93', '95', '91', '184'],
 ['79', '80', '73', '152'],
 ['70', '73', '78', '148'],
 ['93', '89', '96', '192'],
 ['78', '75', '68', '147'],
 ['81', '90', '93', '183'],
 ['88', '92', '86', '177'],
 ['78', '83', '77', '159'],
 ['82', '86', '90', '177'],
 ['86', '82', '89', '175'],
 ['78', '83', '85', '175'],
 ['76', '83', '71', '149'],
 ['96', '93', '95', '192']]
[63.75, 69.5, 87.5, 87.75, 88.25, 92.0, 92.25, 92.25, 94.75, 95.0, 96.0, 99.25, 100.25, 105.25, 107.75, 108.0
, 108.75, 110.75, 111.75, 112.5, 114.75, 115.75, 117.5, 119.0, 122.5]
```

- 평균을 구할때, 현재 주어진 csv파일에서는 각 행마다 데이터가 4개씩 들어가 있지만, 크기가 다른 데이터를 활용할때의 버그를 대비해 4가 아닌, 행의 길이로 나누어줍니다. : length = len(row), average = summation/length