

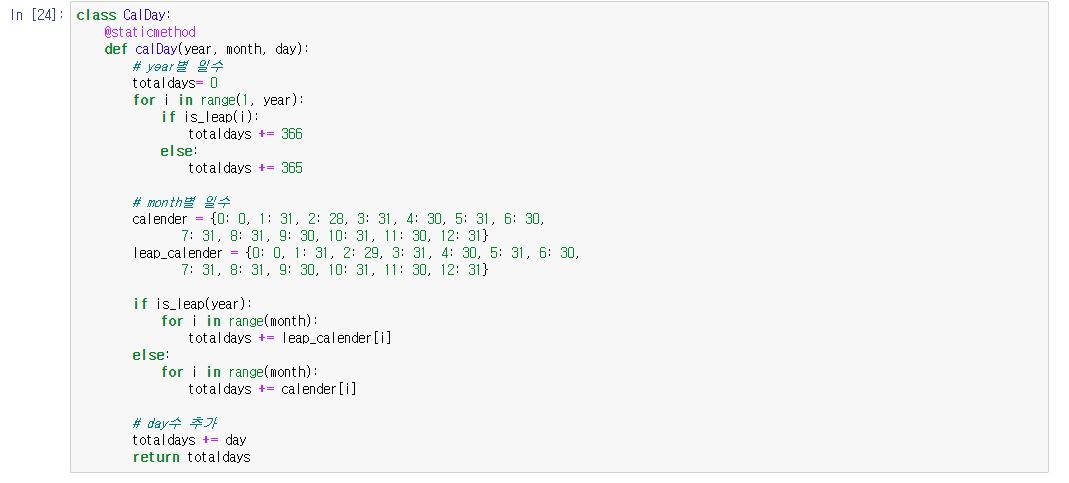
**외부 함수**

input\_date(): year, month, day를 각각 input()으로 입력받아 튜플로 return하는 함수입니다.

is\_leap(): year를 파라미터로 받아 윤년 여부를 확인하는 함수입니다.

checkTime(): time 모듈을 활용하여 파라미터 함수의 실행시간을 측정하는 함수입니다.

**1. 윤년 구하기 프로그램**



**CalDay**: 해당 날짜까지의 총 일수를 계산하는 클래스입니다.

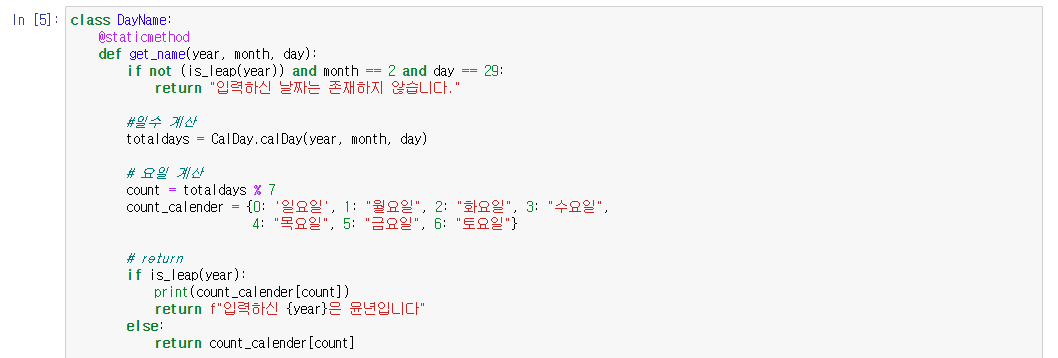
**calDay()**: year, month, day를 인자로 받아 해당 날짜까지의 총 일수를 return하는 함수입니다. staticmethod로 활용합니다.

① year별 일수를 계산합니다. 총 일수를 계산할 totaldays 변수를 초기화하고, 1부터 year-1의 범위동안의 year별 일수를 계산하여 totaldays에 저장합니다. 이때 외부에서 정의한 is\_leap()함수를 활용하여 윤년여부를 판단합니다.

② month별 일수를 계산합니다. 윤년 여부에 따라 0월부터 12월까지의 달력을 calender와 leap\_calender에 각각 사전형으로 저장합니다. 이때 0월은 0일로 저장되어 있습니다. 다음으로, 0월부터 month-1의 범위동안 해당 년도에 맞는 달력을 활용하여 month별 일수를 계산한 다음 totaldays에 추가합니다.

③ day별 일수를 totaldays에 추가합니다.

④ totaldays를 return합니다.



**DayName**: 해당 날짜의 요일을 판별하는 클래스입니다.

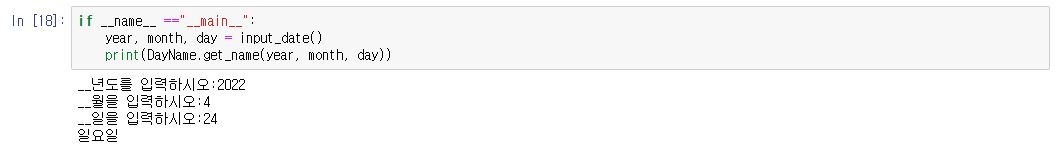
**get\_name()** : year, month, day를 인자로 받아 해당 날짜의 요일을 판별하는 함수입니다. staticmethod로 활용됩니다.

① 윤년이 아닌데 2월 29일을 입력한 경우, 에러 메시지를 return합니다.

② CalDay 클래스의 calDay 메소드를 호출하여 반환값을 totaldays에 저장합니다.

③ 요일별 달력을 만듭니다. totaldays를 7로 나눠 count에 저장하고, 0부터 6의 범위 key값에 일요일부터 토요일까지의 value를 추가한 count\_calender 사전을 만듭니다.

④ 계산한 요일을 출력하거나 return합니다. 윤년인 경우, 해당 날짜의 요일을 print하고 "입력하신 연도는 윤년입니다"라는 메시지를 return합니다. 윤년이 아닌 경우, 해당 날짜의 요일을 return합니다.



윤년 구하기 프로그램을 if \_\_name\_\_ == "\_\_main\_\_"을 통해 실행하여 보았습니다. input\_date()를 실행하여 얻은 값을 year, month, day에 저장하고 DayName의 get\_name() 메소드에 인자로 넣어주었습니다. 2022년 4월 24일에 해당하는 요일이 정상적으로 출력됩니다.

여기까지는 이미 OOP V1에서 구현하였던 프로그램입니다.

**2. MerryChristmas**

인류 역사상 크리스마스는 어떤 요일에 가장 많았을까요? 쉽게 추측할 수는 없지만 아마도 사람들은 매년 토요일이나 일요일이 아닌 평일이 되기를 바라왔을 것입니다. 그러나 실제로 이를 수동으로 확인하기는 매우 어렵습니다. 사람의 손으로 일일이 계산하는 일은 비용도 많이 들 뿐더러 실수가 생길 확률이 높기 때문입니다. 따라서 저는 파이썬 프로그램을 통해 이를 직접 확인해보고자 했습니다.

저의 구현 목표는 다음과 같습니다.

**(1) 연도를 입력하면 해당년도까지 요일별로 크리스마스가 몇 번 있었는지 계산하기**

**(2) (1)의 해당년도까지 D-Day 계산하기. 이때 dday를 변경할 수 있는 기능까지 구현하기**

**(3) 프로그램 실행 날짜로부터 D-Day 계산하기**



**MerryChirstmas**: (1) (2) (3)의 과정을 모두 수행하는 클래스입니다.

**month, day**: 크리스마스 날짜를 가리키는 클래스 변수입니다.

**\_\_init\_\_()**: MerryChristmas의 생성자로, 새 객체가 생성될 때마다 실행되는 매직메소드입니다. year, dday의 값을 인자로 받아와 인스턴스 변수인 self.year, self.\_dday에 저장합니다. 이때 self.\_dday는 이후 property를 통해 보호모드로 접근할 수 있도록 합니다.

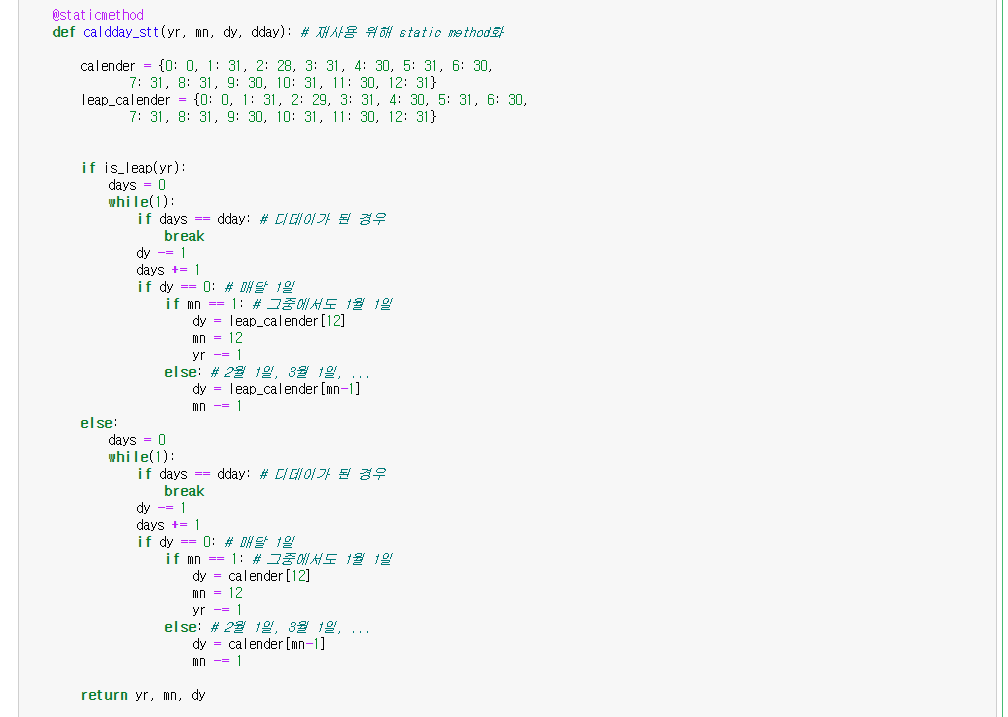
**christmasday()**: 해당년도까지 크리스마스가 요일별로 몇 번 있었는지 확인하는 함수입니다. checkTime의 데코레이터 인자로 활용되어 함수의 실행시간이 측정됩니다.

①totaldays를 0으로 초기화하고, christmas\_calender를 0으로 초기화하여 사전형으로 생성합니다.

②크리스마스의 요일을 구하는 과정입니다. 1부터 self.year까지 for문을 돌면서 크리스마스의totaldays와 count를 계산하고 이를 활용하여 요일의 횟수가 저장된 christmas calender를 업데이트 해줍니다.

③0~6까지 요일의 이름이 저장된 count\_calender를 사전형으로 생성합니다.

④count\_calender와 christmas\_calender를 활용하여 크리스마스의 요일별 횟수를 출력합니다.



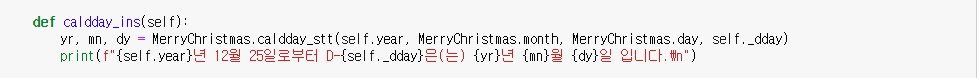
**caldday\_stt()**: yr, mon, day, dday를 인수로 받아 yr, mn, dy로부터의 dday를 구해주는 함수입니다. 나중에 있을 todday()함수에서의 재사용을 위해 staticmethod로 선언해주었습니다.

①calander, leap\_calender: 윤년이 아닌 해와 윤년인 해의 0월부터 12월까지의 일수가 저장된 사전을 생성합니다. 이때 0월은 0일이 저장되도록 생성합니다.

②yr가 윤년인 경우, 반복문의 횟수를 세주는 days를 0으로 초기화합니다. while문을 돌면서 yr년 mn월 dy일에서부터 시작해 점점 dy를 -1씩 해가면서 체크합니다. 매달 1일일 때 leap\_calender를 활용하여 yr, mn, dy를 업데이트해줍니다. 만약 days가 dday와 같다면 날짜를 찾은 것이므로 break합니다.

③yr가 윤년이 아닌 경우 동일한 과정을 leap\_calender 대신 calender를 활용하여 진행합니다.

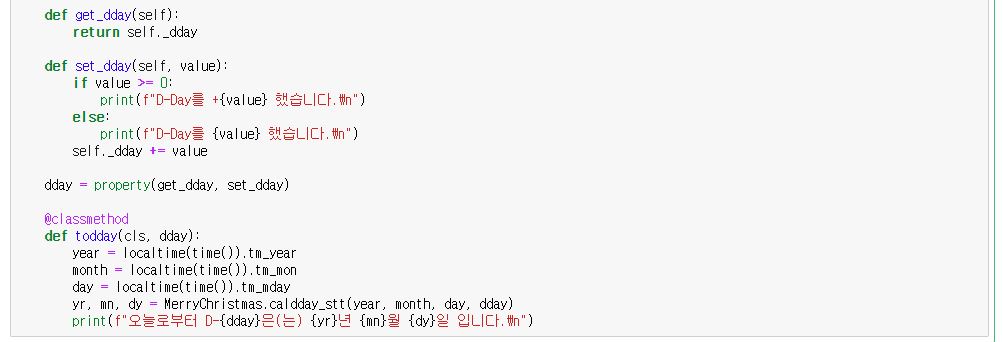
④찾은 yr, mn, dy를 return합니다.



**caldday\_ins()**: 해당년도 크리스마스의 dday를 구해주는 함수입니다.

①self.year, MerryChristmas.month, MerryChristmas.day, self.\_dday를 인자로 caldday\_stt()를 실행합니다.

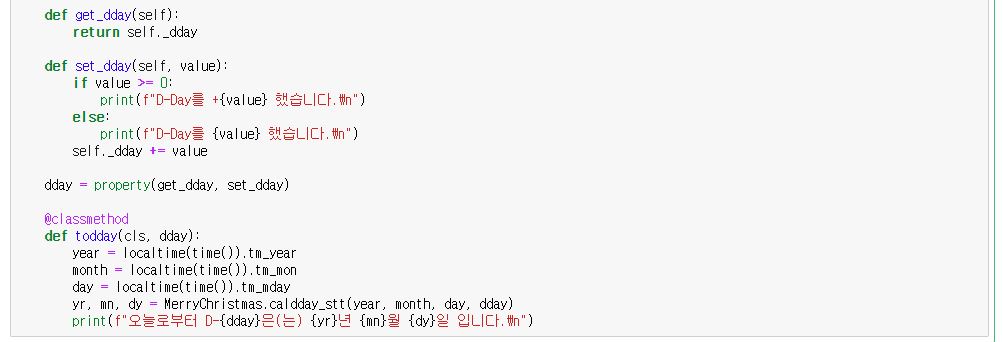
②얻은 값을 yr, mn, dy에 저장하고 정해진 형식에 따라 출력합니다.

****

**get\_dday()**: self.\_dday의 값을 get해주는 함수입니다.

**set\_dday()**: self.\_dday의 값을 set해주는 함수입니다. D-Day를 변경했다는 안내메시지를 출력합니다.

get\_dday와 set\_dday를 활용하여 dday로 property 설정해줍니다.

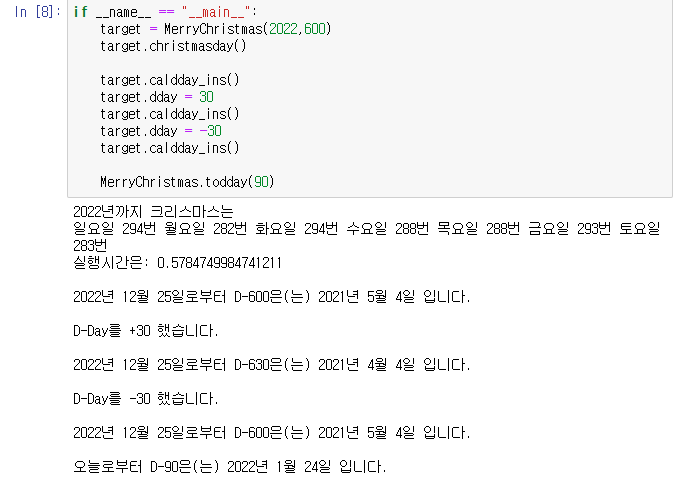
****

**todday()**: 프로그램 실행 날짜로부터 dday를 구해주는 함수입니다. 클래스 메소드로 구현하였습니다.

①localtime과 time 메소드를 활용하여 프로그램 실행 날짜를 추출한 뒤 year, month, day에 저장합니다.

②MerryChristmas 클래스의 caldday\_stt() 메소드를 통해 year, month, day로부터 dday 날짜를 구하고 yr, mn, dy에 저장합니다.

③정해진 형식에 따라 출력합니다.



if \_\_name\_\_ == "\_\_main\_\_":을 통해 직접 실행해 보면 다음과 같습니다.

MerryChristmas(2022, 600)을 객체를 생성하여 target에 저장하고 christmasday() 메소드를 실행합니다. 성공적으로 2022년까지의 크리스마스 요일별 횟수가 출력되는 것을 확인할 수 있습니다. 초기에 프로그램을 작성할 때 떠올렸던 질문의 답을 확인해보면, 그간 인류 역사상 가장 크리스마스가 많았던 요일은 **화요일**과 **일요일**입니다.

christmasday()의 측정시간을 확인해보니 0.578의 시간이 걸린 것을 알 수 있습니다. 2022번의 루프를 돌아야 하기 때문에 시간이 다소 소요되었습니다.

다음으로 caldday\_ins()를 실행했습니다. 2022년 12월 25일로부터 D-600은 2021년 5월 4일입니다. dday에서 30을 더해주었습니다. 2022년 12월 25일로부터 D-630은 2021년 4월 4일입니다. 다시 dday에서 30을 빼주었습니다. 그러자 위에서와 동일한 날짜가 출력됩니다.

마지막으로 todday()를 실행하여 실행 날짜로부터 dday를 구했습니다. 오늘로부터 D-90은 2022년 1월 24일임을 확인할 수 있었습니다.