자료구조 과제 #1

214823 박종현

```python
class Datetime:
    basetime: tuple[int] = (0, 0, 0)
    def __init__(self, year: int, month: int, day: int):
        self.timediff = (year - Datetime.basetime[0]) * 360 + (month - Datetime.basetime[1]) * 30 + (day - Datetime.basetime[2])

    def __str__(self):
        year, month, date = self.date
        return f'Datetime({ year }, { month }, { date })'

    def __repr__(self):
        year, month, date = self.date
        return f'Datetime({ year }, { month }, { date })'

    def __lt__(self, other):
        return self.timediff < other.timediff

    def __eq__(self, other):
        return self.timediff == other.timediff

    def __add__(self, other):
        if type(other) == 'int':
            self.timediff += other
        if type(other) == type(self):
            self.timediff += other.timediff
        return self

    def __sub__(self, other):
        if type(other) == 'int':
            self.timediff -= other
        if type(other) == type(self):
            self.timediff -= other.timediff
        return self

    @property
    def date(self) -> tuple[int]:
        year = self.timediff // 360 + Datetime.basetime[0]
        month = (self.timediff % 360) // 30 + Datetime.basetime[1]
        day = self.timediff % 30 + Datetime.basetime[2]

        if day <= 0:
            day += 30
            month -= 1
        if month <= 0:
            month += 12
            year -= 1
```

```
46
47          return (year, month, day)
48
49  class DateDiff(Datetime):
50      @property
51      def date(self) -> tuple[int]:
52          year = self.timediff // 360 + Datetime.basetime[0]
53          month = (self.timediff % 360) // 30 + Datetime.basetime[1]
54          day = self.timediff % 30 + Datetime.basetime[2]
55
56          return (year, month, day)
57
58      def fromDatetime(datetime: Datetime):
59          _new = DateDiff(-1, -1, -1)
60          _new.timediff = datetime.timediff
61          return _new
62
63  def test():
64      assert Datetime(2020, 1, 1) - Datetime(2019, 1, 1) == Datetime(1, 0, 0)
65      assert Datetime(1900, 12, 30) + Datetime(0, 13, 0) == Datetime(1902, 1, 30)
66      assert DateDiff.fromDatetime(Datetime(1990, 11, 10) - Datetime(1990, 11, 10)) ==
    DateDiff.fromDatetime(Datetime(0, 0, 0))
67
68  if __name__ == '__main__':
69      test()
70
71      '''
72      Expected input: Y1, M1, D1, Y2, M2, D2, Y3, M3, D3
73      0 < Y1, Y3; 0 < M1, M3 <= 12; 0 < D1, D3 <= 30
74      0 <= Y2; 0 <= M2; 0 <= D2
75      '''
76      y1, m1, d1, y2, m2, d2, y3, m3, d3 = map(int, input().split(', '))
77
78      # Validate
79      assert 0 < y1
80      assert 0 < m1 <= 12
81      assert 0 < d1 <= 30
82      assert 0 <= y2
83      assert 0 <= m2
84      assert 0 <= d2
85      assert 0 < y3
86      assert 0 < m3 <= 12
87      assert 0 < d3 <= 30
88
89      print('참' if Datetime(y1, m1, d1) + Datetime(y2, m2, d2) == Datetime(y3, m3, d3) else
    '거짓')
90
```

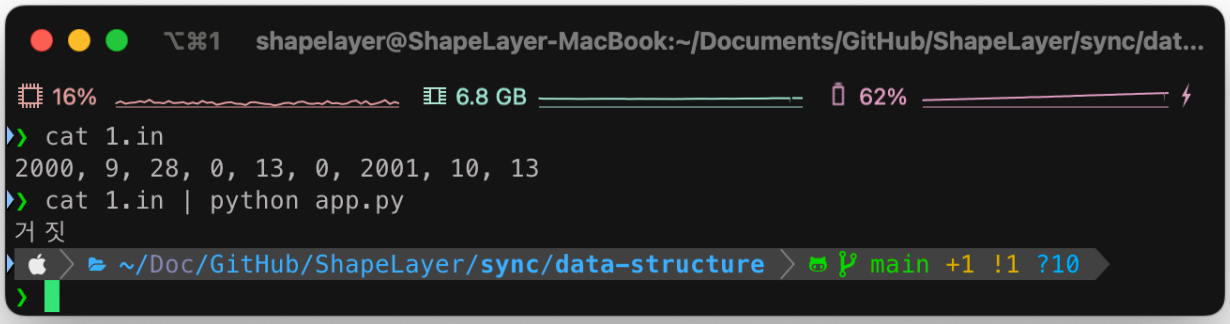
처리 구조

1. 기준 날짜 설정: `Datetime.basetime`
2. 각 `Datetime` 객체는 기준 날짜`Datetime.basetime`로부터 일수 차이 저장: `Datetime.timediff`
3. 날짜 프로퍼티 참조 시 기준 날짜에 일수 차이를 더해 년, 월, 일 생성 후 반환: `@property def date()`
4. 날짜 변화 계산을 위해 덧셈 뺄셈 오버로드
   - 덧셈과 뺄셈은 `self: Datetime`과 `other: [Datetime | int]`를 받음
   - `other`의 타입이 `Datetime`인 경우: 각 객체의 `timediff` 필드끼리 연산
   - `other`의 타입이 `int`인 경우: `self.timediff`와 `other`를 연산

- 연산이 날짜의 차이를 확인하고자 하는 경우, 표현 방식이 수정되어야함
  - `DateDiff(Datetime)` 클래스 도입
  - `DateDiff(Datetime)` 클래스는 년, 월, 일 표현 방식을 조정하기 위해 `@property def date()`를 오버로딩함
  - `DateDiff.fromDatetime(datetime: Datetime)` 메서드: 타입 캐스팅 처리를 위해 작성

실행

예시 1



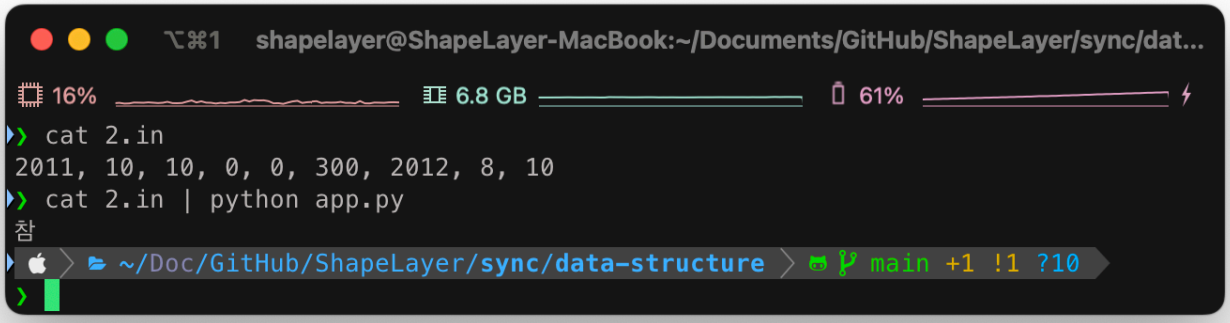
입력

```
1  2000, 9, 28, 0, 13, 0, 2001, 10, 13
```

출력

```
1  거짓
```

예시 2



입력

```
1  2011, 10, 10, 0, 0, 300, 2012, 8, 10
```

출력

```
1  참
```