11/05 DFN 작업상황

I. 편의성 및 가시성 개선

1. 모듈화 완료.

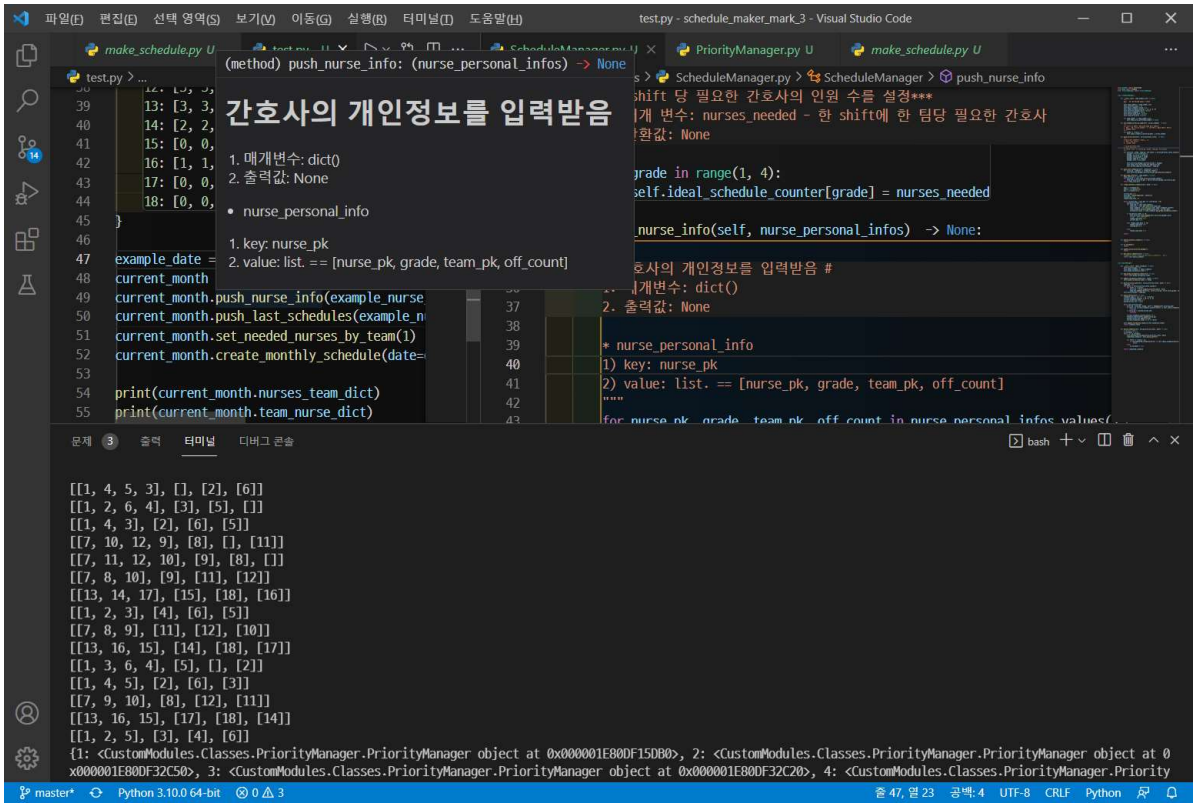
18인 규모의 병동 전체 스케쥴을 7ms 내외로 작성할 수 있습니다. 아래와 같이 직관적으로 스케쥴을 작성할 수 있습니다.

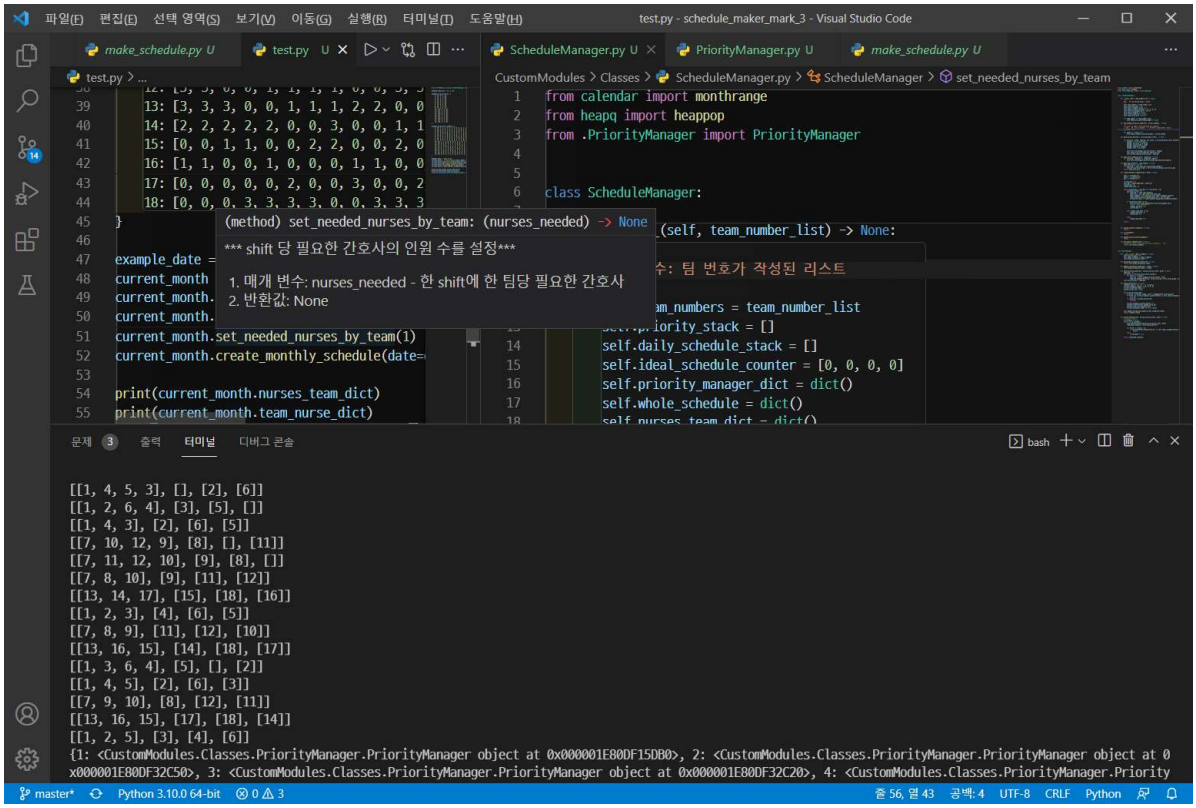
```
current_month = ScheduleManager(team_number_list=example_team_list)
current_month.push_nurse_info(example_nurse_profile)
current_month.push_last_schedules(example_nurse_last_schedule)
current_month.create_monthly_schedule('2021-11-01')
```

최적화 전 (486줄) 보다 코드가 약 70% (현재 174줄) 줄었으며, 가독성도 명료해졌습니다.

2. 독스트링 작성중

메서드별 독스트링을 확인할 수 있습니다.





II. 기능 개선

1. '우선순위' 계산 로직 변경

python은 기본 라이브러리로 최소 힙을 가지고 있다. 따라서 높은 우선순위 == 작은 값 이 되는데, 현재 `compute_priority` 로직은 2000이라는 값에서 `빼는` 형태로 우선도가 배정되어 직관성이 떨어진다.

일단 모든 우선순위 가산점들을 더한 뒤, `음수` 로 바꾸서 출력하는 방향으로 개선.

==> 현재 버그 발생중.. |

수정 후

```
def compute_priority(self, shift, today):

        # 1. 예외 처리 ->
        # 1) None 값 반환하는 경우
        # (1) 근무는 오름차순으로 해야 한다.
        if shift and shift < self.last_shift:
            return None

        # (2) 한 달에 야근을 8회 이상 하지 않는다.
        if shift == 3 and self.monthly_night_shift > 7:
            return None

        # (3) 휴가 관련
        # a. 휴가 전날에 night 근무를 하지 않는다.
        if shift == 3 and today+1 in self.vacation_date:
            return None
```

```
        # b. 휴가날에는 근무를 하지 않는다.
        if shift and today in self.vacation_date:
            return None

        # 2) 최 우선 처리 (우선도 0)
        # (1) 휴가 당일
        if today in self.vacation_date:
            return -5000

        # 2. 우선도 연산 시작.
        priority_token = randrange(500, 50000)

        return -priority_token
```

수정 전

```
def compute_priority(self, shift, today):

        # 1. 예외 처리 ->
        # 1) None 값 반환하는 경우
        # (1) 근무는 오름차순으로 해야 한다.
        if shift and shift < self.last_shift:
            return None

        # (2) 한 달에 야근을 8회 이상 하지 않는다.
        if shift == 3 and self.monthly_night_shift > 7:
            return None

        # (3) 휴가 관련
        # a. 휴가 전날에 night 근무를 하지 않는다.
        if shift == 3 and today+1 in self.vacation_date:
            return None

        # b. 휴가날에는 근무를 하지 않는다.
        if shift and today in self.vacation_date:
            return None

        # 2) 최 우선 처리 (우선도 0)
        # (1) 휴가 당일
        if today in self.vacation_date:
            return 0

        # 2. 우선도 연산 시작.
        priority_token = randrange(800, 1200)
        # 1) 연속 근무
        # (1) shift_streak 2 미만
        # 예상 값: 300 ~ 700
        if shift == self.last_shift and self.shift_streaks < 2:
            priority_token -= randrange(200, 500)

        # (2) shift_streak 2 이상
        # 예상 값: 500 ~ 850
        if shift == self.last_shift and self.shift_streaks < 2:
            priority_token -= randrange(500, 1000)
```

```
        return priority_token
```

2. ScheduleMaanger 클래스 전반적 수정.

`참조 - 역참조를 활용.` 무게가 많이 가벼워졌다.

수정 후 - 클래스 변경

```
class ScheduleManager:

    def __init__(self, team_number_list) -> None:
        # 중략 #
        self.nurses_team_dict = dict()
        self.team_nurse_dict = dict()
```

팀 분할 메서드 추가.

팀별 정보가 필요할 때에는 아래 메서드를 호출.

```
    def get_team_info(self, team_number) -> dict:
        """
        # 팀에 속한 간호사들의 PriorityManager값을 딕셔너리 형태로 반환
        1. 매개변수: int
        2. 출력값: team_info_dict -> dict()
        1) key: nurse_pk
        2) value: PriorityManager object
        """
        team_info_dict = dict()
        for nurse_pk in self.team_nurse_dict[team_number]:
            team_info_dict[nurse_pk] = self.priority_manager_dict[nurse_pk]
        return team_info_dict
```

수정 전

class init 하는 부분. 이처럼 이중 딕셔너리를 만들거나

```
        for team_number in team_number_list:
            self.priority_manager_dict[team_number] = dict()
            self.whole_schedule[team_number] = dict()
```

딕셔너리를 [팀 번호] [간호 번호] 로 이중 참조 하거나

```
    def push_nurse_info(self, nurse_personal_infos)  -> None:

        for nurse_pk, grade, team_pk, off_count in
nurse_personal_infos.values():
            manager = PriorityManager()
            manager.nurse_pk = nurse_pk
            manager.nurse_grade = grade
```

```
            manager.team_pk = team_pk
            manager.offs = off_count

            self.priority_manager_dict[team_pk][nurse_pk] = manager
            self.nurses_team_dict[nurse_pk] = team_pk

    def push_last_schedules(self, schedules):
        for nurse_pk, schedule in schedules.items():
            team_pk = self.nurses_team_dict[nurse_pk]
            self.priority_manager_dict[team_pk][nurse_pk].personalize(schedule)
```

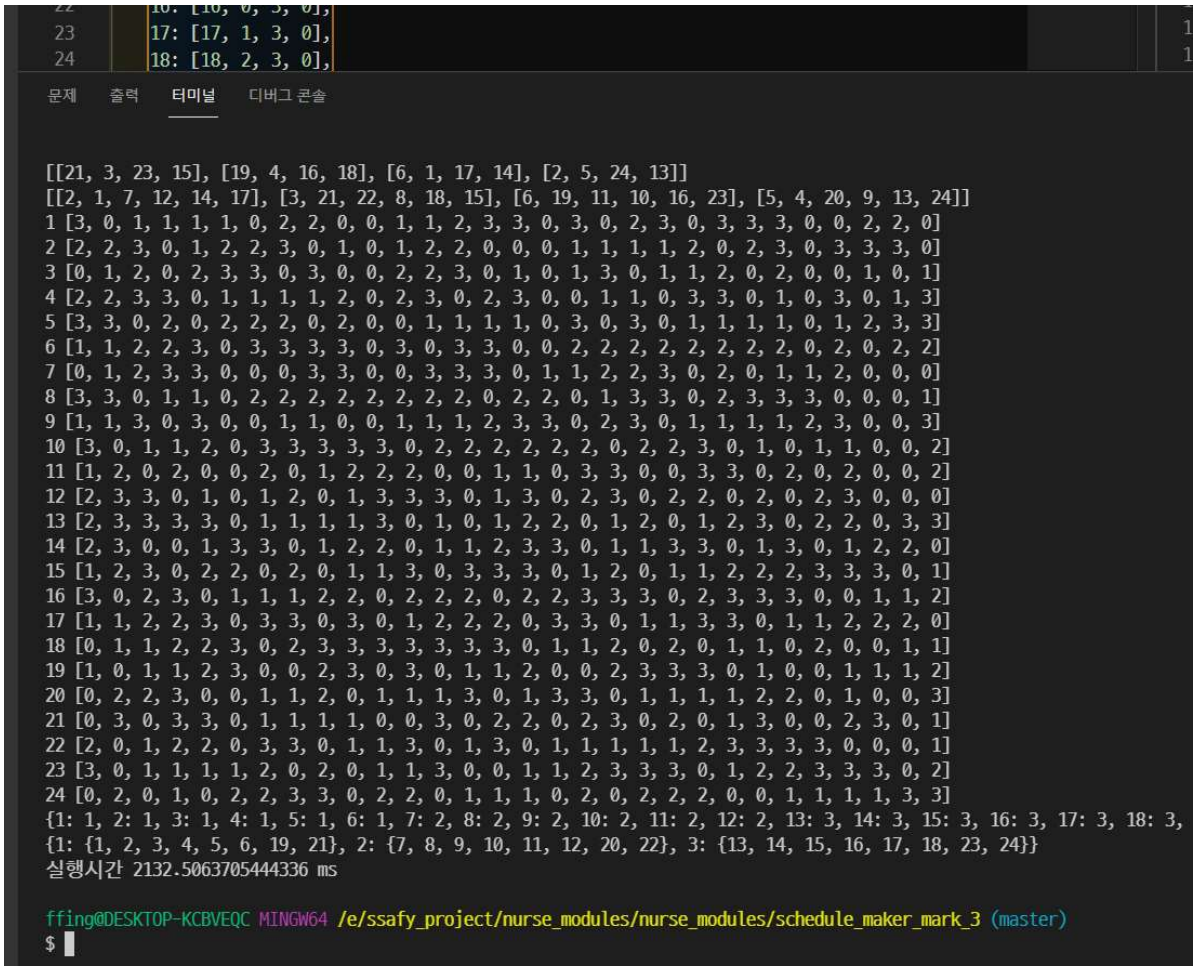
팀별로 분할된 딕셔너리를 재통합하는 함수 등을 유지했었다.

III. 확장성 관련

확장성 테스트 - 실험 완료.

24인 규모, 근무 타입당 팀별 간호사 2명 근무 등도 제작 가능.

랜덤함수 효율 개선 시 더 효율적인 배치 가능 예정.



IV. 기타

heappush 해야할걸 아무 생각 없이 append 했다가 일어난 대형 참사.. 디버깅 두시간 걸림..