새내기 연구 참여 최종 보고서

산업경영공학과

20240442 박선영

소개

-
새내기 연구 참여
교수님 소개

김병인 교수님 bkim@postech.ac.kr

전공분야	물류최적화, OR 응용
연구실	공학4동 217호
Lab	물류 연구실

소개

-
새내기 연구 참여
학부생 소개

박선영 seonyeong@postech.ac.kr

학과 무은재학부

학번 20240442

연구주제

파이썬을 활용한 최적화 문제 해결

물류 최적화 문제는 물류 시스템의 효율성을 극대화하기 위한 최적화 모델을 개발하는 데 중점을 둔다
각 요소의 자원을 최소화하면서도 운영 성과를 극대화하는 방법을 탐구하여, 비용 절감과 운영 속도 향상을 동시에 달성하고자 한다.

교통 신호 최적화 및 자율주행 전용 차선 개발

자율주행 차량 전용 차선을 설정해 수집된 데이터를 활용, 이를 도시 차원으로 확장하여 교통 신호 체계를 최적화한다
도시 내 교통 흐름을 효율적으로 관리할 수 있도록 설계되며 이를 통해 도심 교통의 원활한 운영을 목표로 한다

철강 운송 선박의 효율적 정박 및 운영 체계 구축

포스코의 선박이 최소한의 시간 내에 정박하고 운영될 수 있도록 최적화된 체계를 설계한다
운송 프로세스를 더욱 효율적으로 만들고 전체 물류 비용과 시간을 줄이는 효과를 기대할 수 있다

반도체 소자 설계에서 소자 간 연결 최적화

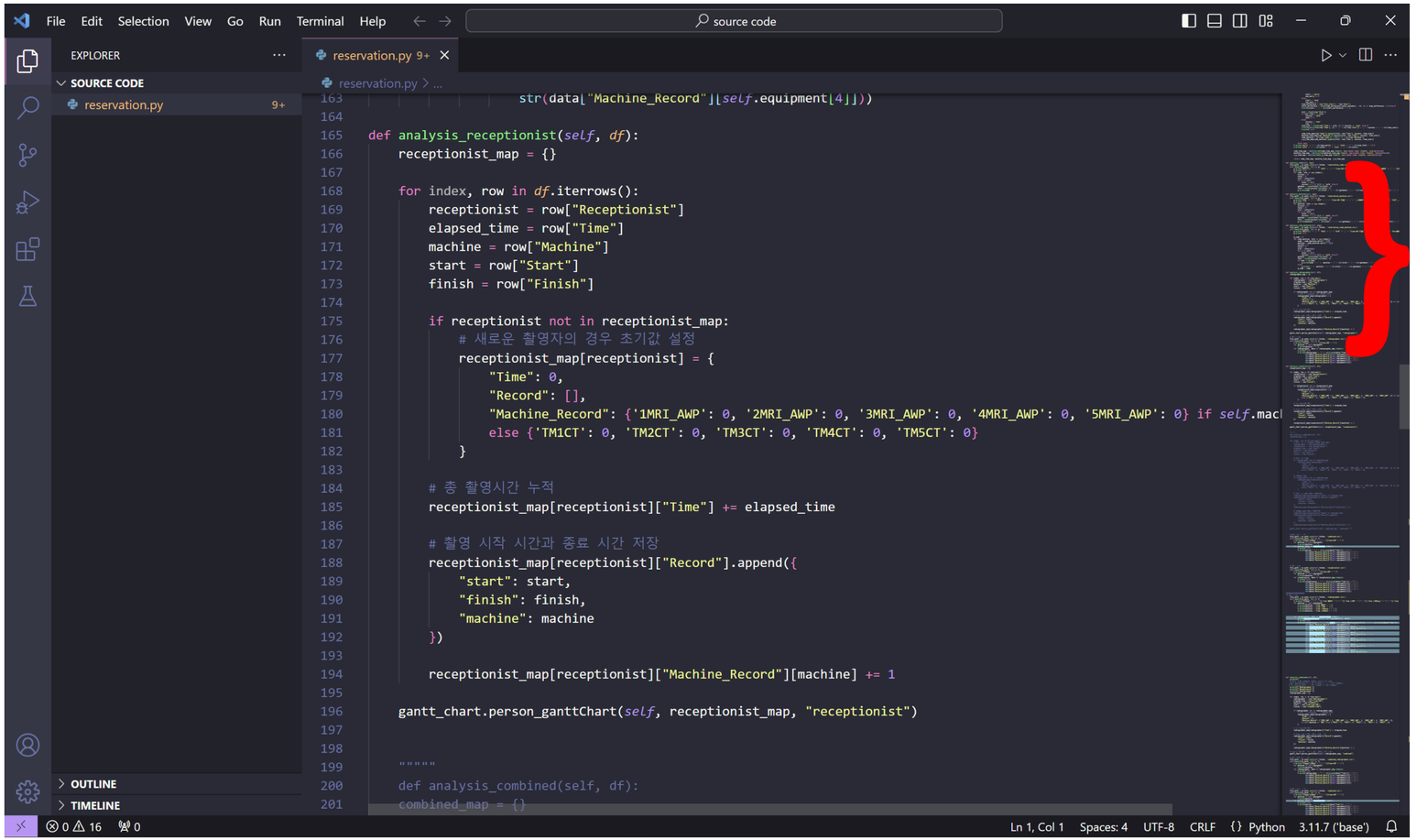
반도체 소자가 가장 작은 단위로 설계될 수 있도록 소자 간의 연결을 최적화
반도체 소자의 집적도를 높이고 성능을 향상시키며 반도체 기술의 발전을 지원하는 것을 목표로 한다

문제상황

- MRI 5대, CT 5대를 예약 알고리즘 대상으로 함 (응급용 제외)
- 처방코드 별로 준비 시간, 검사 시간 등이 다름
- 진료과 별로 선호하는 일정 기준이 다름
  - 내과의 경우 판독이 나와야 결과를 볼 수 있으므로 진료일 당일에 촬영하는 것이 기본임
  - 정형외과의 경우 판독 없이도 진료 가능함
  - 일부 진료과는 담당 의사의 외래 일정에 따라 예약 일정을 생성함
- CT, MRI 모두 07:30~08:30에 병동 환자만 예약 가능함

중간 이전

-촬영 예약 현황표 수정




```
                str(data["Machine_Record"][self.equipment[4]]))

def analysis_receptionist(self, df):
    receptionist_map = {}

    for index, row in df.iterrows():
        receptionist = row["Receptionist"]
        elapsed_time = row["Time"]
        machine = row["Machine"]
        start = row["Start"]
        finish = row["Finish"]

        if receptionist not in receptionist_map:
            # 새로운 촬영자의 경우 초기값 설정
            receptionist_map[receptionist] = {
                "Time": 0,
                "Record": [],
                "Machine_Record": {'1MRI_AWP': 0, '2MRI_AWP': 0, '3MRI_AWP': 0, '4MRI_AWP': 0, '5MRI_AWP': 0} if self.mac
                else {'TM1CT': 0, 'TM2CT': 0, 'TM3CT': 0, 'TM4CT': 0, 'TM5CT': 0}
            }

        # 총 촬영시간 누적
        receptionist_map[receptionist]["Time"] += elapsed_time

        # 촬영 시작 시간과 종료 시간 저장
        receptionist_map[receptionist]["Record"].append({
            "start": start,
            "finish": finish,
            "machine": machine
        })

        receptionist_map[receptionist]["Machine_Record"][machine] += 1

    gantt_chart.person_ganttChart(self, receptionist_map, "receptionist")


    """"""
    def analysis_combined(self, df):
    combined_map = {}
```

기존 코드를 수정하여 Receptionist(촬영 예약자)와 Radiographer(촬영자)를 하나의 대상으로 처리하는 gantt_chart를 만들도록 함.

문제 정의 [input]

예약 현황

장비명	등록번호	처방코드	일자	시작시간	종료시간
3MRI_AWP	00195969	RMNSMR	20240816	9:20	10:20
5MRI_AWP	00326035	RMXMNAE_1	20240816	9:00	9:30
TM2CT	00002421	RCXCABD_1	20240819	8:45	9:00

의료진 근무 현황

일자	성명	진료과	오전	오후
20240816	홍길동	정형외과	N	Y
20240816	의료진2	정형외과	N	N
20240816	의료진3	정형외과	N	N
20240816	의료진4	정형외과	Y	Y
20240816	의료진5	정형외과	N	Y
20240819	홍길동	정형외과	N	Y
20240819	의료진2	정형외과	N	N
20240819	의료진3	정형외과	N	N
20240819	의료진4	정형외과	Y	Y
20240819	의료진5	정형외과	N	Y

장비 목록

장비명	종류
1MRI_AWP	MRI
2MRI_AWP	MRI
3MRI_AWP	MRI
4MRI_AWP	MRI
5MRI_AWP	MRI
TM1CT	CT
TM2CT	CT
TM3CT	CT
TM4CT	CT
TM5CT	CT

장비별 특정 처방코드 촬영시간

장비명	시작시간	종료시간	처방코드
2MRI_AWP	11:00	11:30	RMNSMR
3MRI_AWP	11:15	11:30	RMNSMR
3MRI_AWP	14:00	14:30	RMNSMR
3MRI_AWP	14:30	15:15	RMNSMR
4MRI_AWP	9:00	9:15	RMNSMR
4MRI_AWP	9:30	9:45	RMNSMR
4MRI_AWP	10:00	10:15	RMNSMR
4MRI_AWP	13:30	13:45	RMNSMR
4MRI_AWP	13:45	14:15	RMNSMR
4MRI_AWP	14:15	14:45	RMNSMR
4MRI_AWP	15:00	15:15	RMNSMR
TM5CT	10:30	11:00	RCXCABD1V_1
TM5CT	11:00	12:00	RCXCABD1V_1
TM5CT	10:30	10:45	RCXCABD1V_1
TM5CT	13:30	13:45	RCXCABD1V_1
TM5CT	14:00	14:15	RCXCABD1V_1
TM5CT	14:30	14:45	RCXCABD1V_1
TM5CT	15:00	15:15	RCXCABD1V_1
TM5CT	15:30	15:45	RCXCABD1V_1
TM5CT	16:00	16:15	RCXCABD1V_1

처방코드 별 표준 촬영 시간

처방코드	촬영 시간(분)
RMMR1F	30
RMNSMR	30
RMNSMRC	60
RMNSMRT	30
RMXHE2352_1	60
RMXHE235_1	60
RCXCABAV_1	15
RCXCABA_1	15
RCXCABD1V_1	15
RCXCABD1_1	15
RCXCABD_1	15
RCXCABMD	15

일자별 장비 가용 시간

일자	장비명	시작시간	종료시간
20240816	1MRI_AWP	8:30	19:00
20240816	2MRI_AWP	8:30	19:00
20240816	3MRI_AWP	8:30	19:00
20240816	4MRI_AWP	8:30	19:00
20240816	5MRI_AWP	8:30	19:00
20240816	TM1CT	8:45	18:00
20240816	TM2CT	8:45	18:00
20240816	TM3CT	8:45	18:00
20240816	TM4CT	8:45	18:00
20240816	TM5CT	8:45	18:00

장비별 처방 코드 (촬영가능여부)

처방코드	1MRI_AWP	2MRI_AWP	3MRI_AWP	4MRI_AWP	5MRI_AWP	TM1CT	TM2CT	TM3CT	TM4CT	TM5CT
RMMR1F	1	1	1	1	1	0	0	0	0	0
RMNSMR	0	0	1	1	0	0	0	0	0	0
RMNSMRC	0	0	1	0	0	0	0	0	0	0
RMNSMRT	0	0	1	0	0	0	0	0	0	0
RMXHE2352_1	1	0	0	0	0	0	0	0	0	0
RMXHE235_1	0	1	0	0	0	0	0	0	0	0
RCXCABAV_1	0	0	0	0	0	1	0	0	0	0
RCXCABA_1	0	0	0	0	0	0	1	0	0	0
RCXCABD1V_1	0	0	0	0	0	1	1	1	0	1
RCXCABD1_1	0	0	0	0	0	1	1	1	0	1
RCXCABD_1	0	0	0	0	0	1	1	1	0	1
RCXCABMD	0	0	0	0	0	0	1	0	1	0

중간 이후

-촬영 예약 일시 추천 시스템 구현

중간 이후

-촬영 예약 일시 추천 시스템 구현

과정 [pseudo code]

1. 필요한 라이브러리들 (`sys`, `pandas`, `available_times`)을 임포트합니다.

2. 각 파일에 대한 클래스 정의:
   - ReservationStatus: 예약 데이터 로드 및 저장을 위한 클래스.
   - MedicalStaffSchedule: 의료진 근무 현황을 로드하고 저장하는 클래스.
   - EquipmentAvailabilityByDate: 일자별 장비 가용 시간을 로드하고 저장하는 클래스.
   - EquipmentList: 장비 목록을 로드하고 저장하는 클래스.
   - EquipmentPrescriptionCode: 장비별 처방 코드 (사용 가능 여부)를 로드하고 저장하는 클래스.
   - EquipmentSpecificPrescriptionTime: 장비별 특정 처방 코드 촬영 시간을 로드하고 저장하는 클래스.
   - PrescriptionCodeStandardTime: 처방 코드 표준 촬영 시간을 로드하고 저장하는 클래스.

3. 파일에서 데이터 로드:
   - 각 클래스의 객체를 생성하여 해당 파일 경로를 전달하고, 파일을 로드하여 데이터를 리스트 형태로 저장

4. 파라미터 로드:
   - `load_parameters()` 함수 정의하여 파라미터 파일을 읽고, 선호일, 담당의사, 선호 시작/종료 시간 등을 딕셔너리 형태로 반환

5. 처방 코드에 맞는 장비 목록 필터링:
   - `filter_by_prescription_code()` 함수 정의하여 주어진 처방 코드에 해당하는 장비 목록을 필터링
   - '처방코드'를 인덱스로 설정하고, 사용 가능한 장비만 추출하여 출력

6. 의료진 근무 현황을 반영한 예약 가능한 시간대 필터링:
   - `filter_medical_staff_availability()` 함수 정의하여 의료진의 근무 시간에 맞는 예약 가능한 시간대 리스트를 필터링
   - 오전(9:00~12:00) 및 오후(13:00~18:00) 근무 여부를 체크하고 해당하는 시간대만 필터링

7. 처방 코드에 맞는 표준 촬영 시간 가져오기:
   - `get_standard_time_for_prescription_code()` 함수 정의하여 주어진 처방 코드에 맞는 촬영 시간을 표준 시간 데이터에서 찾아 출력

8. 시간을 분으로 변환:
   - `time_to_minutes()` 함수 정의하여 주어진 시간 형식(문자열, 정수, `datetime.time`)을 분 단위로 변환

9. 예약 가능한 후보일자 생성:
   - `generate_candidate_dates()` 함수 정의하여 예약 가능한 시간대 중에서 후보일자를 생성
   - 선호 시작 시간과 종료 시간을 분으로 변환한 뒤, 예약 가능한 시간대를 필터링하여 후보일자를 추출
   - `candidate_count`만큼 후보일자를 생성하고 반환

10. 장비별 촬영 가능 시간 필터링:
    - `filter_available_equipment_times()` 함수 정의하여 처방 코드에 맞는 장비와 그 장비의 가능한 시간대를 필터링
    - 각 장비의 시작 시간과 종료 시간을 계산하여 시간 차이가 표준 촬영 시간 이상인 경우만 추가

11. 결과를 Excel에 저장:
    - `save_to_excel()` 함수 정의하여 필터링된 장비별 가능한 시간대 데이터를 Excel 파일에 저장
    - 저장할 데이터는 `candidate_count`만큼만 출력되도록 처리

12. 메인 흐름:
    - `load_parameters()`를 통해 파라미터를 로드
    - `filter_by_prescription_code()`를 통해 처방 코드에 맞는 장비 목록을 필터링
    - `filter_medical_staff_availability()`를 통해 의료진 근무 현황을 반영한 예약 가능한 시간대를 필터링
    - `get_standard_time_for_prescription_code()`를 통해 표준 촬영 시간을 가져오기
    - `generate_candidate_dates()`를 통해 후보일자를 생성
    - `filter_available_equipment_times()`로 필터링된 시간대를 출력하고, 그 결과를 Excel에 저장

13. 출력 및 저장:
    - 후보일자와 예약 가능한 장비별 시간대 출력
    - 필터링된 장비별 시간대를 Excel 파일에 저장

중간 이후

-촬영 예약 일시 추천 시스템 구현

과정 [주요 함수]

```python

# 장비별 특정 처방코드 촬영 시간 파일
class EquipmentSpecificPrescriptionTime:
    def __init__(self, file_path):
        self.data = self.load_data(file_path)

    def load_data(self, file_path):
        df = pd.read_excel(file_path)
        # 장비 코드, 시작 시간, 종료 시간, 촬영 종류로 리스트 생성
        return df.values.tolist()

# 처방코드 별 표준 촬영 시간 파일
class PrescriptionCodeStandardTime:
    def __init__(self, file_path):
        self.data = self.load_data(file_path)

    def load_data(self, file_path):
        df = pd.read_excel(file_path)
        # 처방 코드와 표준 촬영 시간으로 리스트 생성
        return df.values.tolist()

# 객체 생성
reservation_status = ReservationStatus(r"C:\Users\user\Desktop\선영\산경과 새연참\중간 이후\MRI data format (241104)\input\
medical_staff_schedule = MedicalStaffSchedule(r"C:\Users\user\Desktop\선영\산경과 새연참\중간 이후\MRI data format (241104)
equipment_availability_by_date = EquipmentAvailabilityByDate(r"C:\Users\user\Desktop\선영\산경과 새연참\중간 이후\MRI data
equipment_list = EquipmentList(r"C:\Users\user\Desktop\선영\산경과 새연참\중간 이후\MRI data format (241104)\input\장비목록.
equipment_prescription_code = EquipmentPrescriptionCode(r"C:\Users\user\Desktop\선영\산경과 새연참\중간 이후\MRI data forma
equipment_specific_prescription_time = EquipmentSpecificPrescriptionTime(r"C:\Users\user\Desktop\선영\산경과 새연참\중간 이
prescription_code_standard_time = PrescriptionCodeStandardTime(r"C:\Users\user\Desktop\선영\산경과 새연참\중간 이후\MRI dat

```

Input 엑셀파일을 불러와 읽고 사용하기 쉽도록 객체로 생성.

```python
def load_parameters(file_path):
    try:
        df = pd.read_excel(file_path, header=None)
        parameters = {
            "처방코드": df.iloc[1, 1],  # 2행 2열 (처방코드)
            "담당의사": df.iloc[2, 1],  # 3행 2열 (담당의사)
            "예약타입": df.iloc[3, 1],  # 4행 2열 (예약 타입)
            "선호일": df.iloc[4, 1],  # 5행 2열 (선호일)
            "선호시작시간": df.iloc[5, 1],  # 6행 2열 (선호 시작 시간)
            "선호종료시간": df.iloc[6, 1],  # 7행 2열 (선호 종료 시간)
            "후보일자개수": df.iloc[11, 1],  # 12행 2열 (후보일자 개수)
        }
        return parameters  # 파라미터를 딕셔너리 형태로 반환
    except Exception as e:
        print(f"파일 로드 중 오류 발생: {e}")
        return None
```

Parameter 파일을 행-열 별로 불러와 저장하고 예외 처리.

```python
# 처방코드에 맞는 장비 목록을 필터링하는 함수
def filter_by_prescription_code(prescription_code, prescription_code_df):
    # 처방코드를 행 인덱스로 설정
    prescription_code_df.set_index('처방코드', inplace=True)

    # 처방코드가 있는지 확인
    if prescription_code in prescription_code_df.index:
        # 처방코드가 1인 열을 찾아서 장비 목록 추출
        available_equipment = prescription_code_df.loc[prescription_code]  # 해당 처방코드 행을 추출
        available_equipment = available_equipment[available_equipment == 1].index.tolist()  # 1인 열을 찾아 장비명만 추출
    else:
        print(f"처방코드 '{prescription_code}'를 찾을 수 없습니다.")
        return []

    # 처방코드에 맞는 장비 목록 출력
    print(f"처방코드 '{prescription_code}'에 맞는 장비 목록:")
    print(available_equipment)

    return available_equipment


if parameters:
    prescription_code = parameters["처방코드"]
    filter_by_prescription_code(prescription_code, prescription_code_df)
```

Parameter 파일의 처방코드를 다룰 수 있는 장비를 찾아 목록으로 만들고 저장.

중간 이후

-촬영 예약 일시 추천 시스템 구현

과정 [주요 함수]

```python
# 의료진 근무 현황을 반영하여 예약 가능한 시간대를 계산
def filter_medical_staff_availability(available_slots, staff_schedule_df, parameters, date):
    medical_staff_name = parameters["담당의사"]  # 파라미터에서 담당 의사 이름 가져오기

    # 의료진 근무 현황에서 해당 의료진의 근무 여부 확인
    staff_schedule = staff_schedule_df[staff_schedule_df["성명"] == medical_staff_name]

    if staff_schedule.empty:
        print(f"의료진 '{medical_staff_name}'에 대한 근무 정보가 없습니다.")
        return []

    # 근무 가능한 날짜에 해당하는 데이터 필터링
    work_schedule = staff_schedule[staff_schedule["일자"] == date]

    if work_schedule.empty:
        print(f"의료진 '{medical_staff_name}'의 '{date}'에 대한 근무 정보가 없습니다.")
        return []
    # 오전(9:00-12:00), 오후(13:00-18:00) 근무 여부 확인
    is_morning = work_schedule["오전"].values[0] == "Y"
    is_afternoon = work_schedule["오후"].values[0] == "Y"
    # 근무 가능한 시간대 계산
    available_slots_filtered = []
    for slot_start, slot_end in available_slots:
        if is_morning and 9 * 60 <= slot_start < 12 * 60:
            available_slots_filtered.append((slot_start, min(slot_end, 12 * 60)))
        if is_afternoon and 13 * 60 <= slot_start < 18 * 60:
            available_slots_filtered.append((max(slot_start, 13 * 60), min(slot_end, 18 * 60)))
    # 근무 시간 출력
    print(f"의료진 '{medical_staff_name}'의 근무 가능 시간:")
    if is_morning:
        print("- 오전: 9:00-12:00")
    if is_afternoon:
        print("- 오후: 13:00-18:00")
    if not is_morning and not is_afternoon:
        print("- 근무 불가능")

    return available_slots_filtered
```

Parameter 상 지정 의료진의 근무현황을 불러와 의료진의 예약 가능한 시간대를 일자별로 확인 후 저장

```python
# 파라미터에서 처방코드를 추출하여 available_equipment 리턴
if parameters:
    prescription_code = parameters["처방코드"]
    available_equipment = filter_by_prescription_code(prescription_code, prescription_code_df)

    # 장비별 특정 처방코드 촬영 시간 엑셀 파일 로드
    equipment_specific_prescription_df = load_equipment_specific_prescription_time(equipment_specific_prescription_file_p

    # 파라미터에서 얻은 표준 촬영 시간 (분) 추출
    time_required = get_standard_time_for_prescription_code(prescription_code, standard_time_df)

    # 해당 처방코드에 맞는 장비별 촬영 가능한 시간대 출력 (시간 차이가 time_required 이상인 경우만 필터링)
    available_times = filter_available_equipment_times(prescription_code, available_equipment, equipment_specific_prescri


    for i, (equipment, start_time, end_time) in enumerate(available_times):
        if i >= parameters["후보일자개수"]:  # candidate_count만큼만 출력
            break
        print(f"장비: {equipment}, ({start_time}, {end_time})")
        save_to_excel(available_times, r"C:\Users\user\Desktop\선영\산경과 새연참\중간 이후\MRI data format (241104)\output
    else:
        print(f"처방코드 '{prescription_code}'에 대해 {time_required}분 이상의 시간대가 없습니다.")
```

-파라미터 상 예약 타입 (최대한 빠른 시일 내에 예약 후 촬영 or 지정 의료진의 근무표 상 의료진과의 상담과 가까운 날짜에 예약 후 촬영) 에 따라 두 경우로 처리.

-파라미터 상 지정 의료진의 근무표를 참고하여 근무하는 시간대일 것.

-파라미터 상 처방코드를 다룰 수 있는 장비일 것.

-예약 현황 파일을 참고하여 현재 예약되어 있는 일자, 시간 대의 장비가 아닐 것.

-장비별 처리 가능한 처방코드 별 시간대에 포함되어 있을 것.

-처방코드 별 촬영에 소요되는 표준 촬영 시간 이상의 범위가 확보될 것.

-파라미터 상 환자의 선호일과 선호 시간 이후의 시간대일 것.

등을 고려하여 파라미터 상의 후보일자 개수만큼 촬영 가능한 장비와 시작, 종료 시간을 엑셀파일로 저장.

중간 이후

-촬영 예약 일시 추천 시스템 구현

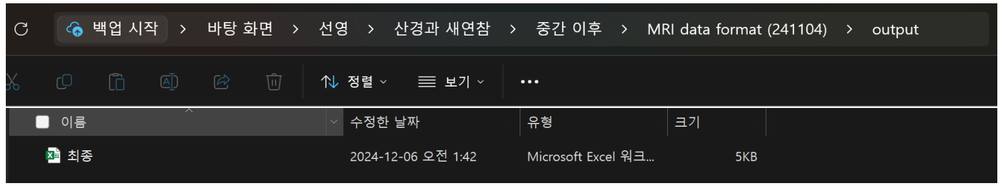
결과

```
처방코드 'RMNSMR'에 맞는 장비 목록:
['3MRI_AWP', '4MRI_AWP']
의료진 '홍길동'의 근무 가능 시간:
- 오전: 9:00-12:00
- 오후: 13:00-18:00
필터링된 예약 가능 시간대: [(540, 720)]
처방코드 'RMNSMR'에 대한 표준 촬영 시간: 30분
추천 후보일자:
처방코드 'RMNSMR'에 맞는 장비 목록:
['3MRI_AWP', '4MRI_AWP']
처방코드 'RMNSMR'에 대한 장비별 가능한 시간대:
장비: 3MRI_AWP, 시작시간: 09:00:00, 종료시간: 15:00:00
장비: 4MRI_AWP, 시작시간: 09:00:00, 종료시간: 15:00:00
처방코드 'RMNSMR'에 맞는 장비 목록:
['3MRI_AWP', '4MRI_AWP']
처방코드 'RMNSMR'에 대한 표준 촬영 시간: 30분
PS C:\Users\user\Downloads\MRI 코드 (241206)>
```

최종 후보 일자를 선정하기까지 고려할 사항들을 올바르게 처리하고 있는지 확인하고자 일련의 과정을 출력하며 확인.
마지막 세 줄의 장비명이 후보일자 개수 (파라미터 파일 상 3)에 맞는 장비 명과 (시작시간, 종료시간) 임.

최종 예약 후보일자 (3개) 만을 저장한 엑셀파일.
경로에 맞는 output 폴더에 저장됨.

장비	시작시간	종료시간
3MRI_AWP	9:00	9:30
4MRI_AWP	9:00	9:30
3MRI_AWP	9:30	10:00




```python
result_data=[]

for j in range(num2):
    #print(f"장비: {}, ({minutes_to_time(temp_list[j][1])}, {minutes_to_time(temp_list[j][2])})")
    result_data.append([temp_list[j][0], minutes_to_time(temp_list[j][1]), minutes_to_time(temp_list[j][2])])
# DataFrame으로 변환
    result_df = pd.DataFrame(result_data, columns=['장비', '시작시간', '종료시간'])

result_df.to_excel(r"최종.xlsx", index=False)
```

소감

파이썬을 활용한 의료촬영장비의 효율적 예약 시스템 구현

이 코드는 여러 엑셀 파일에서 예약 현황, 의료진 근무 현황, 장비 가용 시간 등을 불러와 객체 형태로 필요한 데이터를 처리한다. 주어진 처방코드에 맞는 장비 목록을 필터링하고, 의료진의 근무 현황을 반영하여 예약 가능한 시간대만 추출한다. 사용자가 지정한 선호 시작 시간과 종료 시간에 맞는 후보일자를 생성하고, 각 장비의 촬영 가능한 시간을 계산하여 필터링한다. 필터링된 예약 가능 시간대와 장비별 가능한 시간대를 출력하며, 이 데이터를 엑셀 파일에 저장한다. 결과적으로, candidate_count만큼만 출력하며, 해당 데이터는 최종적으로 엑셀 파일로 저장된다.

소감

이번 과제를 진행하면서 처음에는 500줄에 가까운 파이썬 코드를 작성하는 것에 많은 어려움을 겪었다. 특히 후보일자를 도출해내는 과정에서 여러 데이터 소스를 처리해야 했고, 이를 하나의 논리적 흐름으로 엮어가는 것이 상당히 까다로웠다. 데이터의 양과 종류가 많아 코드의 구조나 판단 과정을 구상하는 데에 어려움을 겪었지만, input 파일 처리 과정을 하나하나 출력하여 확인하면서 해결해 나갔다. 그 과정에서 코드의 각 기능이 제대로 작동하는지 확인하고 수정할 수 있었던 점에서 큰 보람을 느꼈다. 한편으로는 오류가 발생했을 때 이를 해결하는 과정에서 시간이 많이 소요되었고, 코드가 길어져 변수와 함수의 이름이 너무 길어져서 관리가 힘든 점도 있었다. 파이썬을 잘 다루지 못하는 편이라 문법을 다시 공부해야 했고, 그로 인해 과제 완료까지 많은 시간을 투자하게 되었지만, 결국 완성된 코드와 결과물을 얻을 수 있어서 매우 뜻깊었다.
코드를 작성하는 과정에서 가장 잘했다고 느낀 점은 문제를 해결하기 위한 논리적 접근을 꾸준히 수정하며, 최종적으로 코드의 목적을 충실히 구현해낸 것이다. 많은 자료형과 다양한 데이터를 처리하는 과정에서 효율적인 방법을 생각하고 이를 코드로 옮기는데 중점을 두었으며 각 데이터를 처리하는 함수들을 단계적으로 작성하여 최종적인 목표를 향해 차근차근 나아갈 수 있었다. 오류가 발생할 때마다 이를 해결하려는 끈기와 과정의 재구성이 중요한 부분임을 깨닫고, 그 과정에서 더욱 발전할 수 있었다. 코드가 잘 작동하는 모습을 보고, 내가 해결한 문제들을 하나씩 처리해 나간 결과물이 나와서 매우 뿌듯함을 느꼈다.