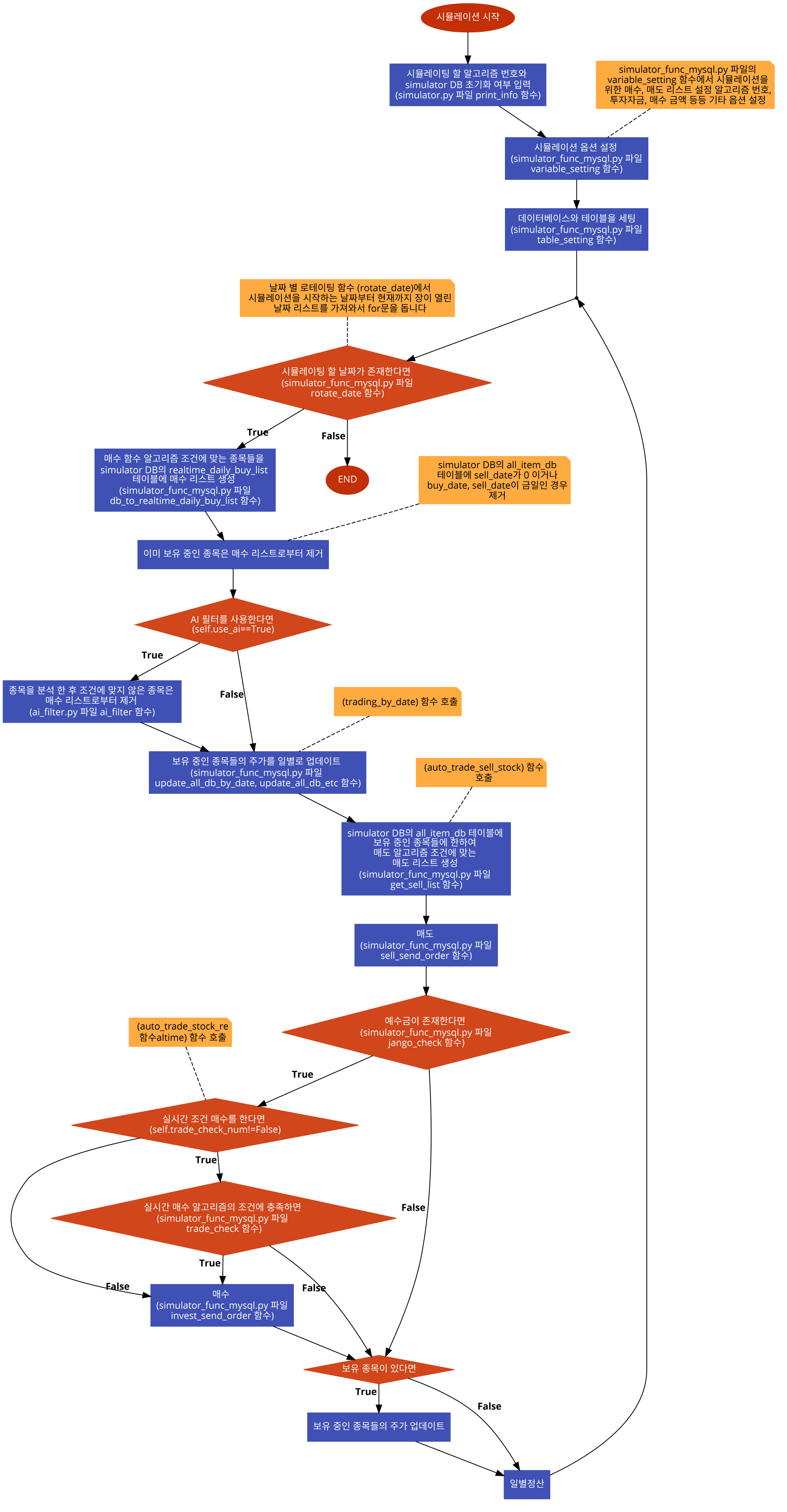
시뮬레이션 시작
시뮬레이팅 할 알고리즘 번호와 simulator DB 초기화 여부 입력 (simulator.py 파일 print_info 함수)
simulator_func_mysql.py 파일의 variable_setting 함수에서 시뮬레이션을 위한 매수, 매도 리스트 설정 알고리즘 번호, 투자자금, 매수 금액 등등 기타 옵션 설정
시뮬레이션 옵션 설정 (simulator_func_mysql.py 파일 variable_setting 함수)
데이터베이스와 테이블을 세팅 (simulator_func_mysql.py 파일 table_setting 함수)
날짜 별 로테이팅 함수 (rotate_date)에서 시뮬레이션을 시작하는 날짜부터 현재까지 장이 열린 날짜 리스트를 가져와서 for문을 돕니다
시뮬레이팅 할 날짜가 존재한다면 (simulator_func_mysql.py 파일 rotate_date 함수)
True
False
END
매수 함수 알고리즘 조건에 맞는 종목들을 simulator DB의 realtime_daily_buy_list 테이블에 매수 리스트 생성 (simulator_func_mysql.py 파일 db_to_realtime_daily_buy_list 함수)
simulator DB의 all_item_db 테이블에 sell_date가 0 이거나 buy_date, sell_date이 금일인 경우 제거
이미 보유 중인 종목은 매수 리스트로부터 제거
AI 필터를 사용한다면 (self.use_ai==True)
True
False
종목을 분석 한 후 조건에 맞지 않은 종목은 매수 리스트로부터 제거 (ai_filter.py 파일 ai_filter 함수)
(trading_by_date) 함수 호출
보유 중인 종목들의 주가를 일별로 업데이트 (simulator_func_mysql.py 파일 update_all_db_by_date, update_all_db_etc 함수)
(auto_trade_sell_stock) 함수 호출
simulator DB의 all_item_db 테이블에 보유 중인 종목들에 한하여 매도 알고리즘 조건에 맞는 매도 리스트 생성 (simulator_func_mysql.py 파일 get_sell_list 함수)
매도 (simulator_func_mysql.py 파일 sell_send_order 함수)
예수금이 존재한다면 (simulator_func_mysql.py 파일 jango_check 함수)
(auto_trade_stock_re 함수altime) 함수 호출
True
실시간 조건 매수를 한다면 (self.trade_check_num!=False)
True
실시간 매수 알고리즘의 조건에 충족하면 (simulator_func_mysql.py 파일 trade_check 함수)
False
True
False
False
매수 (simulator_func_mysql.py 파일 invest_send_order 함수)
보유 종목이 있다면
True
False
보유 중인 종목들의 주가 업데이트
일별정산