



# META INVESTMENT

FULL AUTO TRADING 진아람

# INDEX

1. Team 소개

2. 주제선정

3. 구현목표

4. DATASET

5. DEVELOPING TOOL

6. 자동화 시스템

7. AUTO Trading

8. ARIMA

9. MACHINE LEARNING

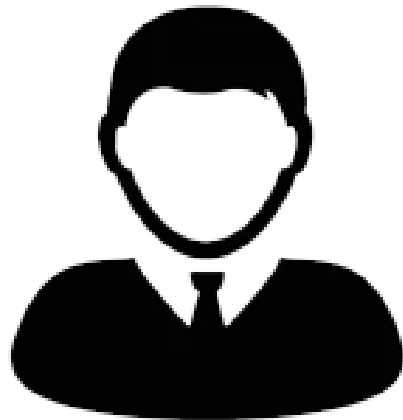
10. DEVELOPING PROCESS

11. TIME LINE

# 팀 소개



팀장 진아람



신명재



이재호



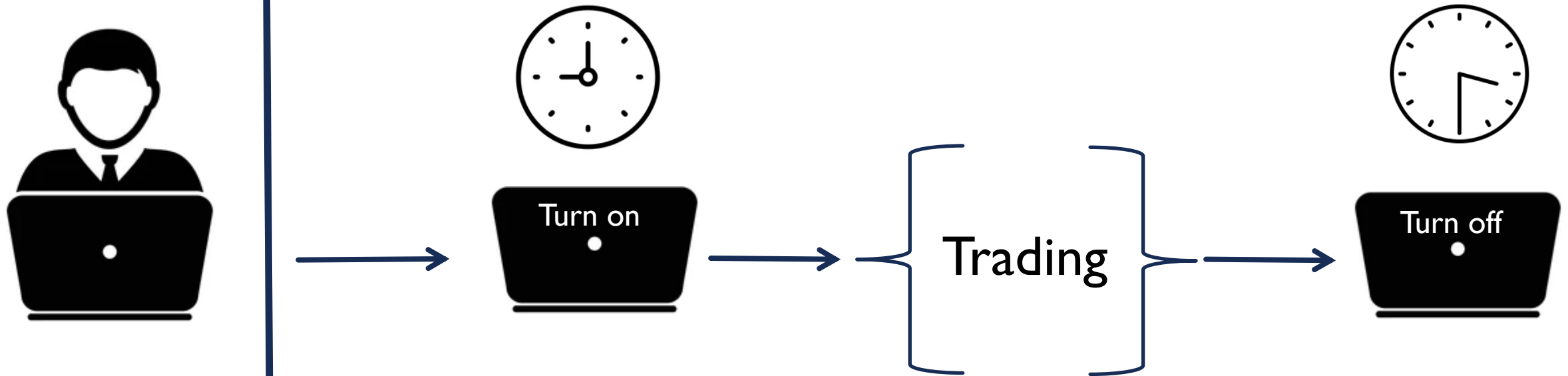
김은탁

# 주제 선정



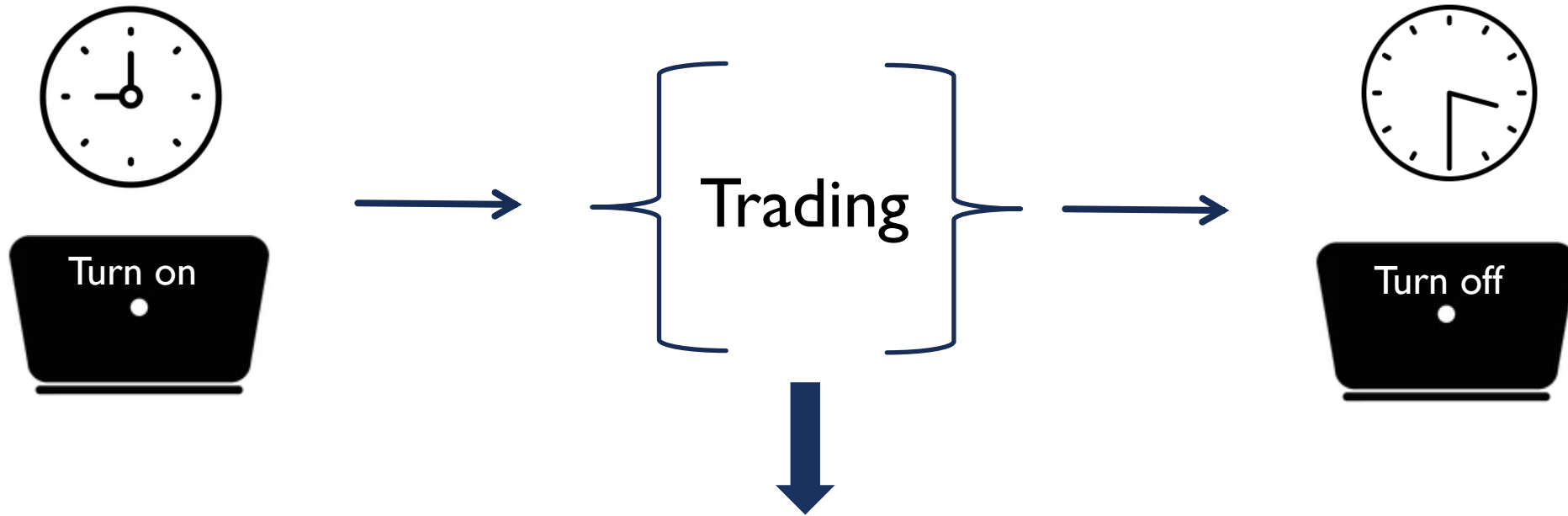
1. 개인 업무로 인하여 매매를 원할 히 할 수 없는 상황을 보완
2. 지속된 뇌동매매를 극복
3. 과다한 욕심을 배제한 규격화된 거래
4. 작더라도 꾸준한 수익

# 구현목표



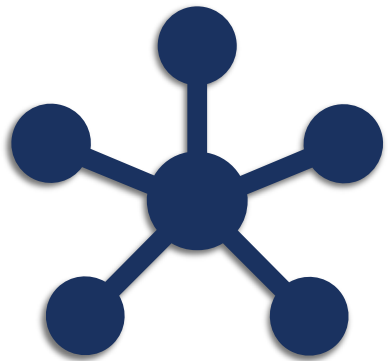
**Full AUTO Trading**

# 구현목표



## Trading Algorithm

1. 수익구간 ( $0\% < \text{수익} < 5\%$ )
2. 손실구간 ( $-5\% < \text{손실} < 0\%$ )



자동화 Algorithm



AI 모델



링크

기술 요구 사항

# DATASET



영웅문4

제공환경

PC 프로그램(WINDOWS 7 서비스팩1 이상)

거래가능상품

국내주식

ETF/ETN/ELW

선물옵션

펀드

해외주식

FnGuide

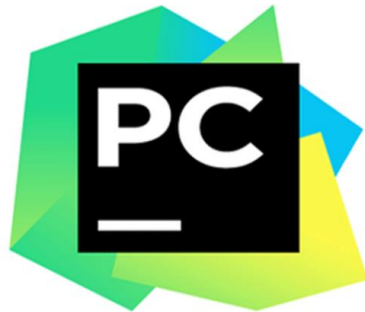




# DEVELOPING TOOL PART.1





TensorFlow



# DEVELOPING TOOL PART.2

키움증권  +

 KOALoader.dll  
 KOASudioSA



키움증권  KIWOOM OPEN API  
키움 Open API

고객ID   
비밀번호   
인증비밀번호

고객 아이디 저장 ☒ 모의투자접속 ☒

ID와 비밀번호를 입력해 주십시오.

 +  +

MACHINE LEARNING PLUS   
All about **ARIMA**  
**FORECASTING**  
BUILD AR, MA, ARIMA, SARIMAX AND AUTOARIMA FROM SCRATCH

# CORE TECHNOLOGY

## Machine Learning



CatBoost

***XGBoost***

*Ensemble*



Class  
 $f(x)$

## Trading



KOAStrudioSA



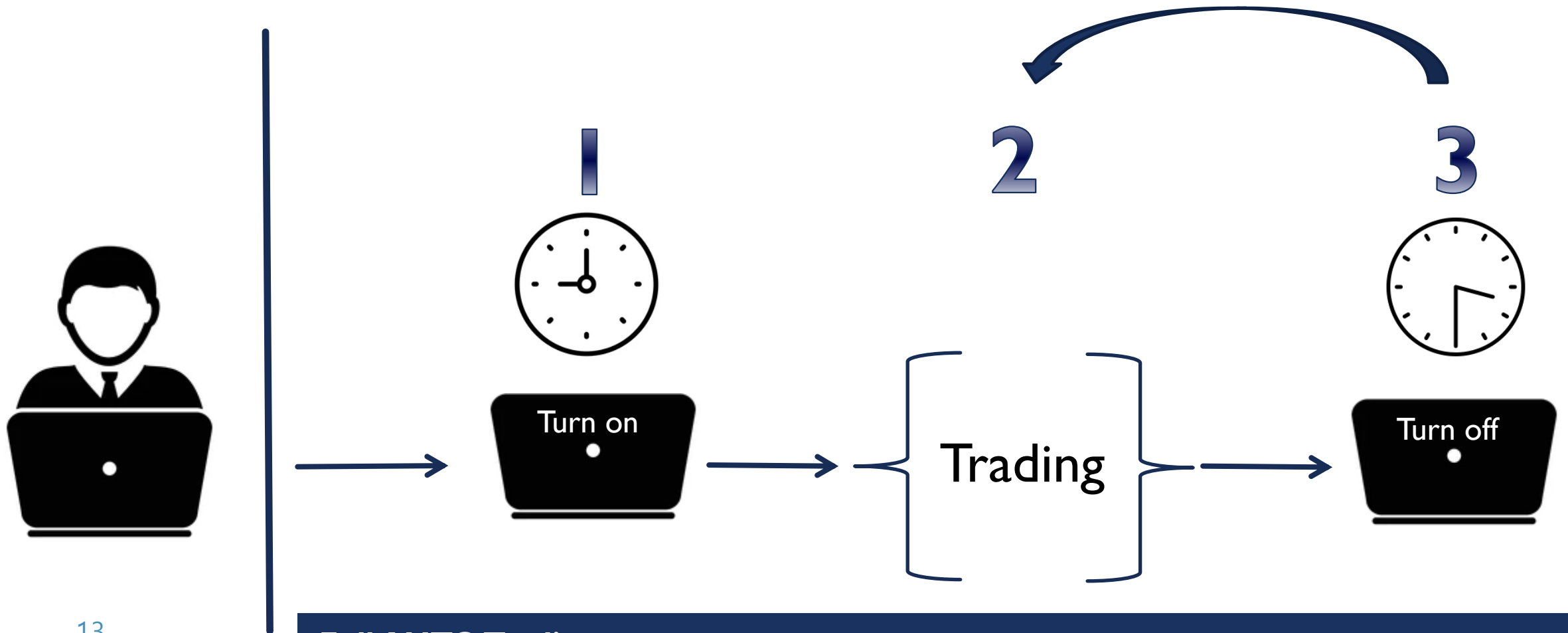
## Visual

**matplotlib**  
Version 3.2.1

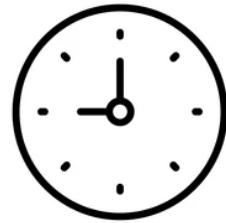
# 자동화 시스템 SET UP

1. 로그인 : 작업 스케줄러 Application과 + OPEN API 를 이용한 자동 로그인
2. 종목검색 : KIWOOM OPEN API와 PY QT BASE ALGORITHM을 이용한 종목 검색
3. 실시간 거래: KIWOOM OPEN API와 PY QT BASE ALGORITHM을 이용한 종목 검색
4. 장종료: 다시 종목 검색으로 2번으로 돌아가 다시 동일 작업 시행

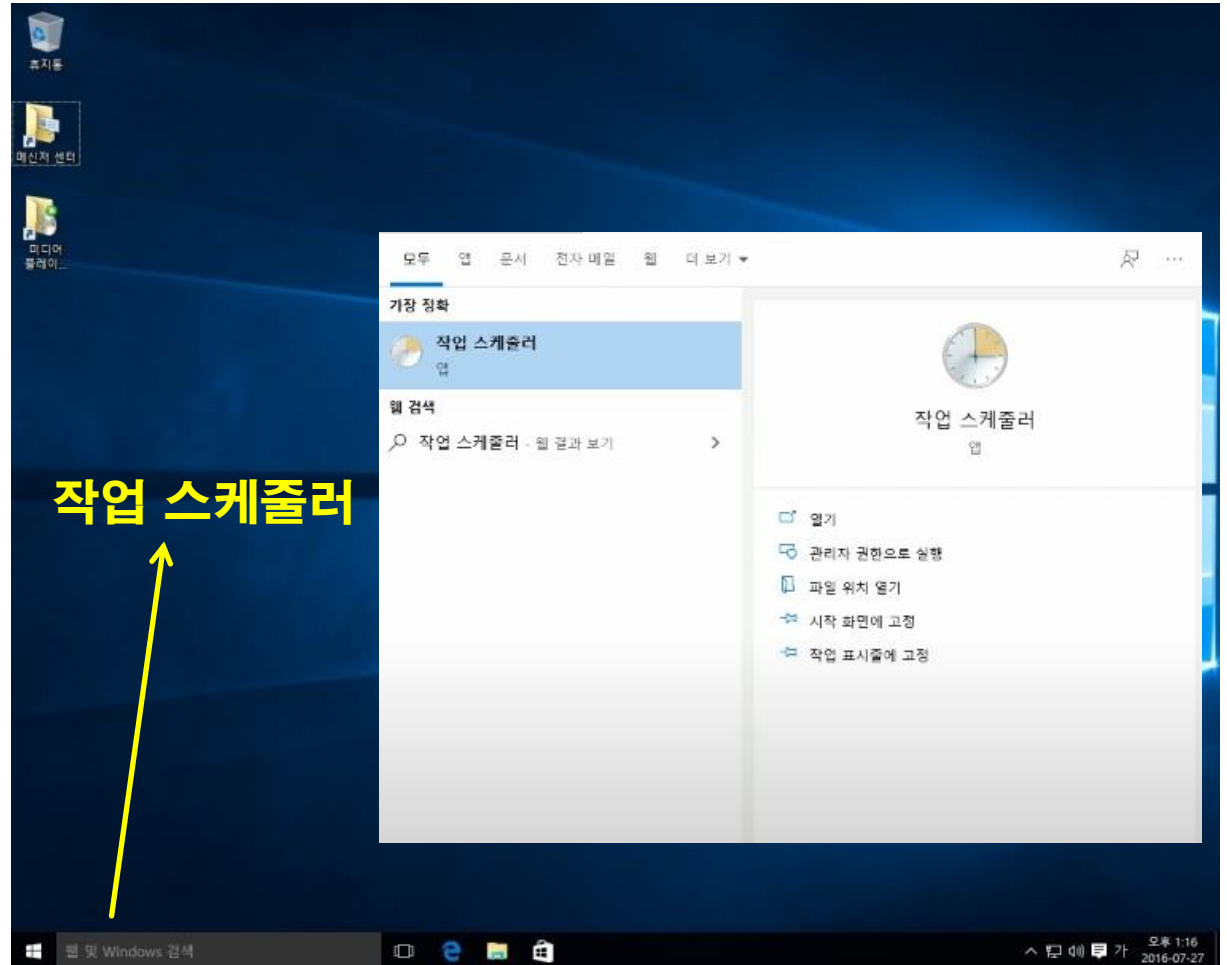
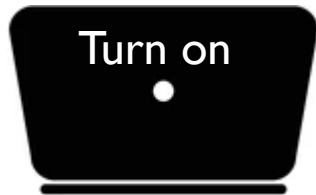
# 1. 자동화 시스템 SET UP



# 자동화 시스템: 로그인 및 프로그램 실행



Turn on



# 자동화 시스템: 로그인 및 프로그램 실행

새 작업 만들기

일반 트리거 동작 조건 설정

이름(M): META INVESTMENT ACCESS

위치: ₩

만든 아: DESKTOP-H2UGBDA#bit

설명(D): 이것은 METAINVESTMENT FAT PROGRAM 실행 알람

보안 옵션

작업을 실행할 때 사용할 사용자 계정:  
DESKTOP-H2UGBDA#bit 사용자 또는 그룹 변경(U)...

☐ 사용자가 로그인할 때만 실행(R)

☒ 사용자의 로그인 여부에 관계없이 실행(W)

☐ 암호를 저장하지 않습니다. 이 작업에서는 로컬 컴퓨터 리소스에만 액세스할 수 있습니다(P).

☒ 가장 높은 수준의 권한으로 실행(I)

☐ 숨김(E) 구성 대상(C): Windows Vista™, Windows Server™ 2008

확인 취소

새 트리거 만들기

작업 시작(G): 예약 상태

설정

☐ 한 번(N)

☒ 매일(D)

☐ 매주(W)

☐ 매월(M)

시작(S): 2022-04-04 오전 8:45:00

표준 시간대 간 동기화(Z)

매(C): 1 일마다

고급 설정

☐ 작업이 지연되는 최대 시간(임의 지연)(K): 1 시간

☐ 작업 반복 간격(P): 1 시간 기간(F): 1 일

☐ 반복 기간이 종료될 때 실행 중인 모든 작업 중지(I)

☐ 다음 기간 이상 실행되는 작업 중지(L): 3 일

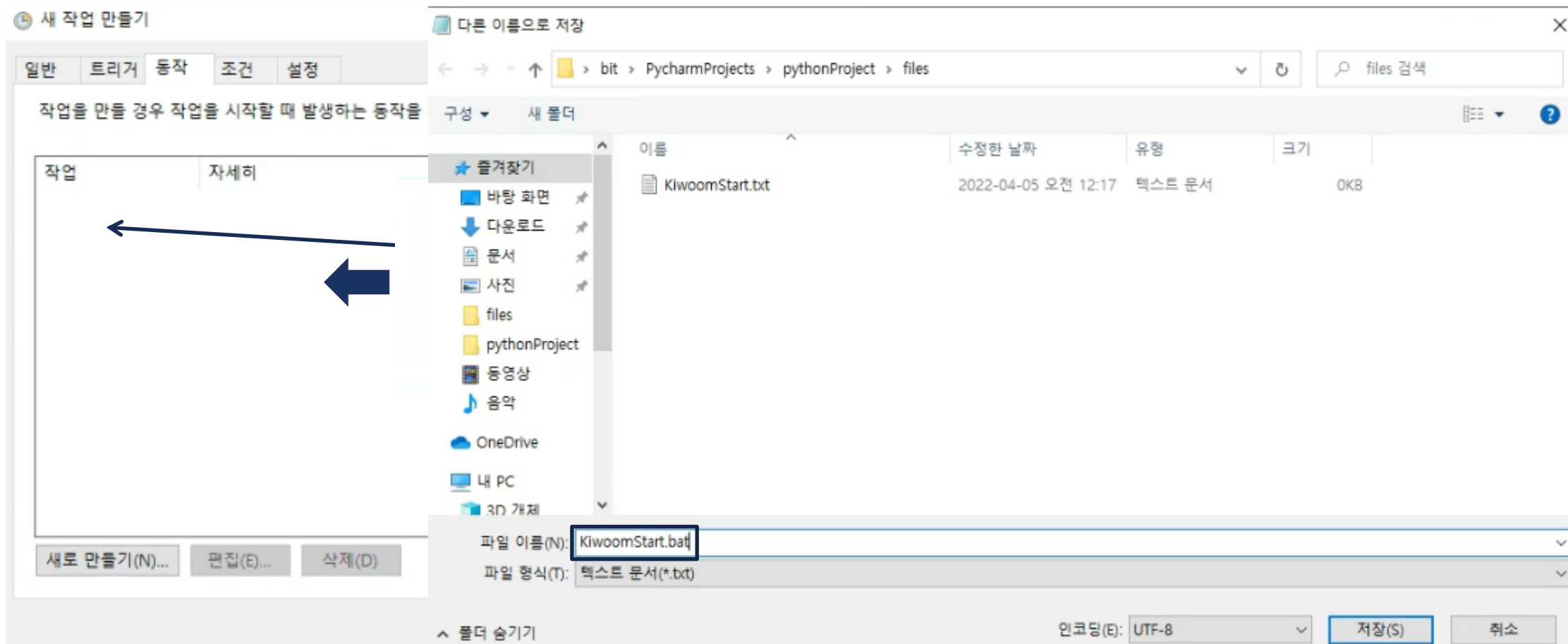
☐ 만료(X): 2023-04-04 오후 11:58:42

표준 시간대 간 동기화(E)

☒ 사용(B)

확인 취소

# 자동화 시스템: 로그인 및 프로그램 실행





# 자동화 시스템:로그인 및 프로그램 실행

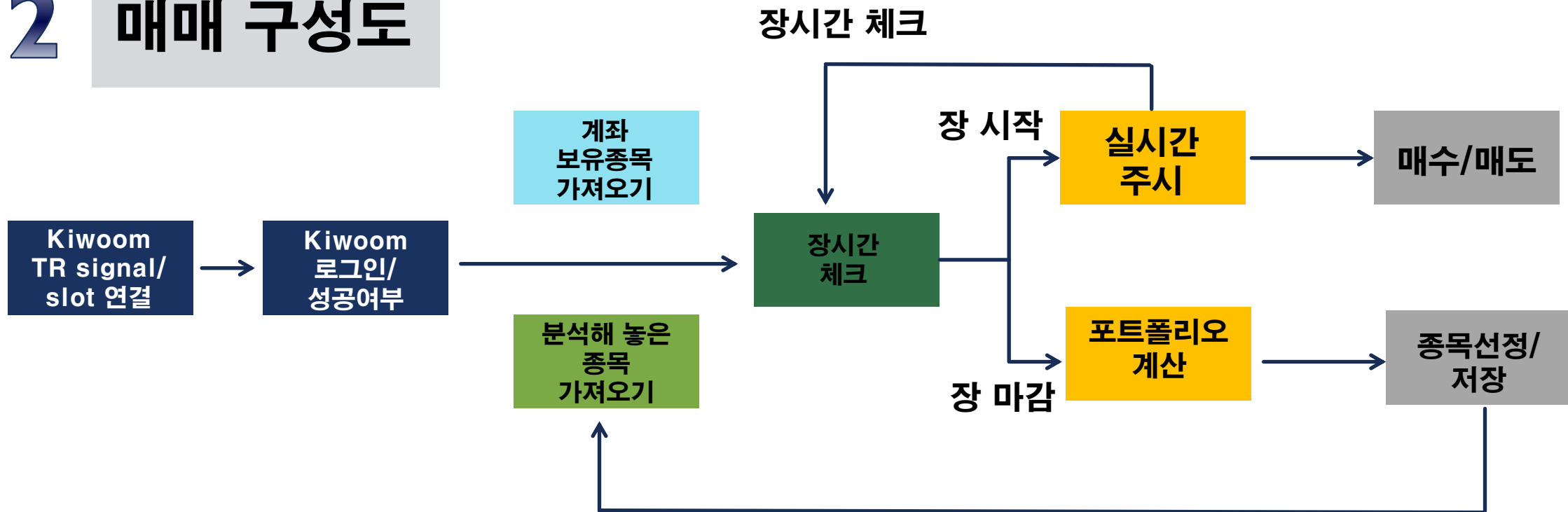


```
C:\Users\bit\PycharmProjects\pythonProject>title Kiwoom Start
C:\Users\bit\PycharmProjects\pythonProject>cd C:\Users\bit\PycharmProjects\pythonProject
C:\Users\bit\PycharmProjects\pythonProject>call activate Finproject
(Finproject) C:\Users\bit\PycharmProjects\pythonProject>python __init__.py
실행할 메인 클래스
Ji_class 입니다
<iwoom 클래스 입니다.

[GetPCIdentity] VER 3.2.0.0 build 2015.8.12
[GetPCIdentity] VER 3.2.0.0 build 2015.8.12
('OP_ERR_NONE', '정상처리')
나의 보유 계좌번호 8018605611
예수금을 요청하는 부분
스크린: 2000, 요청이름: 예수금상세현황요청, tr코드: opw00001 --- [100000] 모의투자 조회완료
예수금 <class 'str'>
예수금 형변환 91687526
출금가능금액 000000091230290
출금가능금액 형변환 91230290
계좌평가 잔고내역 요청하기 연속조회 0
스크린: 2000, 요청이름: 계좌평가잔고내역요청, tr코드: opw00018 --- [100000] 모의투자 조회완료
총매입금액 8242190
```

# AUTO TRADING: API 를 이용한 ALGORITHM

## 2 매매 구성도



# AUTO TRADING : FUNCTION CALLBACK FROM API

**TR 목록**

- opt10001 : 주식기본정보요청
- opt10002 : 주식거래원요청
- opt10003 : 체결정보요청
- opt10004 : 주식호가요청
- opt10005 : 주식일주환시분요청
- OPT10006 : 주식시분요청
- opt10007 : 시세표상정보요청
- opt10008 : 주식외국인요청
- OPT10009 : 주식기관요청
- OPT10010 : 업종프로그램요청
- opt10011 : 신규인수권전체시세요청
- opt10012 : 주문제결요청
- opt10013 : 신용매매동향요청
- opt10014 : 공매도추이요청
- opt10015 : 일별거래상세요청
- OPT10016 : 신고저가요청
- opt10017 : 상하한가요청
- OPT10018 : 고저가근접요청
- opt10019 : 가격급등락요청
- OPT10020 : 호가잔량상위요청
- OPT10021 : 호가잔량중요요청
- OPT10022 : 잔량돌출중요요청
- OPT10023 : 거래량급중요요청
- OPT10024 : 거래량강신요청
- OPT10025 : 매물대집중요청
- opt10026 : 고저PER요청
- opt10027 : 전일대비등락률상위요청
- opt10028 : 시가대비등락률요청
- opt10029 : 예상제결등락률상위요청
- opt10030 : 당일거래량상위요청
- OPT10031 : 전일거래량상위요청
- OPT10032 : 거래대금상위요청
- opt10033 : 신용비율상위요청
- OPT10034 : 외국인간별매매상위요청
- OPT10035 : 외국인연속매매상위요청
- opt10036 : 외국인한도소진률증가상위
- opt10037 : 외국계장구매상위요청
- opt10038 : 종목별증권사순위요청
- OPT10039 : 증권사별매매상위요청
- opt10040 : 당일주요거래원요청
- opt10041 : 조기종료통화단위요청
- opt10042 : 순매수거래원순위요청
- opt10043 : 거래원매물대분석요청
- OPT10044 : 일별기관매매종목요청
- opt10045 : 종목별기관매매추이요청
- opt10046 : 체결강도추이일별요청
- opt10047 : 체결강도추이일별요청
- OPT10048 : ELW일별민감도지표요청

**[SendOrder() 함수]**

```
SendOrder(  
    BSTR sRQName, // 사용자 구분명  
    BSTR sScreenNo, // 화면번호  
    BSTR sAccNo, // 계좌번호 10자리  
    LONG nOrderType, // 주문유형 1:신규매수, 2:신규매도, 3:매수취소, 4:매도취소, 5:매수정정, 6:매도정정  
    LONG nQty, // 종목코드 (6자리)  
    LONG nPrice, // 주문수량  
    BSTR sHogaGb, // 거래구분(혹은 호가구분)은 아래 참고  
    BSTR sOrgOrderNo // 원주문번호, 신규주문에는 공백 입력, 정정/취소시 입력합니다.  
)
```

서버에 주문을 전송하는 함수입니다.  
9개 인자값을 가진 주식주문 함수이며 리턴값이 0이면 성공이며 나머지는 에러입니다.  
1호에 5회만 주문가능하며 그 이상 주문요청하면 에러 -308을 리턴합니다.  
※ 시장가주문시 주문가격은 0으로 입력합니다. 주문가능수량은 해당 종목의 상한가 기준으로 계산됩니다.  
※ 취소주문일때 주문가격은 0으로 입력합니다.

**[거래구분]**

- 00 : 지정가
- 03 : 시장가
- 05 : 조건부지정가
- 06 : 최유리지정가
- 07 : 최우선지정가
- 10 : 지정가IOC
- 13 : 시장가IOC
- 16 : 최유리IOC
- 20 : 지정가FOK
- 23 : 시장가FOK
- 26 : 최유리FOK
- 61 : 장전시간외증가
- 62 : 시간외단일가매매
- 81 : 장후시간외증가

※ 요일투자에서는 지정가 주문과 시장가 주문만 가능합니다.

**[정규장 외 주문]**

장전 통시호가 주문

- 08:30 ~ 09:00. 거래구분 00:지정가/03:시장가 (일반주문처럼)
- ※ 08:20 ~ 08:30 시간의 주문은 처음에서 대기하여 08:30 에 순서대로 거래소로 전송합니다.

장전시간외 증가

- 08:30 ~ 08:40. 거래구분 61:장전시간외증가, 가격 0입력
- ※ 전일 증가로 거래, 미체결시 자동취소되지 않음

장마감 통시호가 주문

- 15:20 ~ 15:30. 거래구분 00:지정가/03:시장가 (일반주문처럼)
- 장후 시간외 증가
- 15:40 ~ 16:00. 거래구분 81:장후시간외증가, 가격 0입력
- ※ 당일 증가로 거래

**출력**

여기에 전문 조회 데이터가 표시됩니다.

실시간목록 TR 목록 종목정보 개발가이드 화면목록

조회 데이터 실시간 데이터 조회한 TR목록

# AUTO TRADING : FUNCTION CALLBACK FROM API

파일(F) 보기(V) 도움말(H) TR 목록

다음

TR 목록

- opt10001 : 주식기본정보요청
- opt10002 : 주식거래원요청
- opt10003 : 체결정보요청
- opt10004 : 주식호가요청
- opt10005 : 주식일주환시분요청
- OPT10006 : 주식시분요청
- opt10007 : 시세표상정보요청
- opt10008 : 주식외국인요청
- OPT10009 : 주식기관요청
- OPT10010 : 업종프로그램요청
- opt10011 : 신규인수권전체시세요청
- opt10012 : 주문제결요청
- opt10013 : 신용매매동향요청
- opt10014 : 공매도추이요청
- opt10015 : 일별거래상세요청
- OPT10016 : 신고저가요청
- opt10017 : 상하한가요청
- OPT10018 : 고저가근접요청
- opt10019 : 가격급등락요청
- OPT10020 : 호가잔량상위요청
- OPT10021 : 호가잔량중요요청
- OPT10022 : 잔량돌출중요요청
- OPT10023 : 거래량급중요요청
- OPT10024 : 거래량경신요청
- OPT10025 : 매물대집중요청
- opt10026 : 고저PER요청
- opt10027 : 전일대비등락상위요청
- opt10028 : 시가대비등락상위요청
- opt10029 : 예상제결등락상위요청
- opt10030 : 당일거래량상위요청
- OPT10031 : 전일거래량상위요청
- OPT10032 : 거래대금상위요청
- opt10033 : 신용비율상위요청
- OPT10034 : 외국인간별매매상위요청
- OPT10035 : 외국인연속매매상위요청
- opt10036 : 외국인한도소진율증가상위
- opt10037 : 외국계장구매상위요청
- opt10038 : 종목별증권사순위요청
- OPT10039 : 증권사별매매상위요청
- opt10040 : 당일주요거래원요청
- opt10041 : 조기종료통화단위요청
- opt10042 : 순매수거래원순위요청
- opt10043 : 거래원매물대분석요청
- OPT10044 : 일별기관매매종목요청
- opt10045 : 종목별기관매매추이요청
- opt10046 : 체결강도추이일별요청
- opt10047 : 체결강도추이일별요청

[SendOrder() 함수]

```
SendOrder(  
    BSTR sRQName, // 사용자 구분명  
    BSTR sScreenNo, // 화면번호  
    BSTR sAccNo, // 계좌번호 10자리  
    LONG nOrderType, // 주문유형 1:신규매수, 2:신규매도, 3:매수취소, 4:매도취소, 5:매수정정, 6:매도정정  
    BSTR sCode, // 종목코드 (6자리)  
    LONG nQty, // 주문수량  
    LONG nPrice, // 주문가격  
    BSTR sHogaGb, // 거래구분(혹은 호가구분)은 아래 참고  
    BSTR sOrgOrderNo // 원주문번호, 신규주문에는 공백 입력, 정정/취소시 입력합니다.  
)
```

서버에 주문을 전송하는 함수입니다.  
9개 인자값을 가진 주식주문 함수이며 리턴값이 0이면 성공이며 나머지는 에러입니다.  
1호에 5회만 주문가능하며 그 이상 주문요청하면 에러 -308을 리턴합니다.  
※ 시장가주문시 주문가격은 0으로 입력합니다. 주문가능수량은 해당 종목의 상한가 기준으로 계산됩니다.  
※ 취소주문일때 주문가격은 0으로 입력합니다.

[거래구분]  
00 : 지정가  
03 : 시장가  
05 : 조건부지정가  
06 : 최유리지정가  
07 : 최우선지정가  
10 : 지정가IOC  
13 : 시장가IOC  
16 : 최유리IOC  
20 : 지정가FOK  
23 : 시장가FOK  
26 : 최유리FOK  
61 : 장전시간외증가  
62 : 시간외단일가매매  
81 : 장후시간외증가  
※ 요일투자에서는 지정가 주문과 시장가 주문만 가능합니다.

[정규장 외 주문]  
장전 동시호가 주문  
08:30 ~ 09:00. 거래구분 00:지정가/03:시장가 (일반주문처럼)  
※ 08:20 ~ 08:30 시간의 주문은 처음에서 대기하여 08:30 에 순서대로 거래소로 전송합니다.  
장전시간외 증가  
08:30 ~ 08:40. 거래구분 61:장전시간외증가, 가격 0입력  
※ 전일 증가로 거래, 미체결시 자동취소되지 않음  
장마감 동시호가 주문  
15:20 ~ 15:30. 거래구분 00:지정가/03:시장가 (일반주문처럼)  
장후 시간외 증가  
15:40 ~ 16:00. 거래구분 81:장후시간외증가, 가격 0입력  
※ 당일 증가로 거래

출력  
여기에 전문 조회 데이터가 표시됩니다.

실시간목록 TR 목록 종목정보 개발가이드 화면목록

조회 데이터 실시간 데이터 조회한 TR목록

# AUTO TRADING : FUNCTION CALLBACK FROM API

[SendOrder() 함수]

```
SendOrder(  
  BSTR sRQName, // 사용자 구분명  
  BSTR sScreenNo, // 화면번호  
  BSTR sAccNo, // 계좌번호 10자리  
  LONG nOrderType, // 주문유형 1:신규매수, 2:신규매도 3:매수취소, 4:매도취소, 5:매수정정, 6:매도정정  
  BSTR sCode, // 종목코드 (6자리)  
  LONG nQty, // 주문수량  
  LONG nPrice, // 주문가격  
  BSTR sHogaGb, // 거래구분(혹은 호가구분)은 아래 참고  
  BSTR sOrgOrderNo // 원주문번호. 신규주문에는 공백 입력, 정정/취소시 입력합니다.  
)
```

서버에 주문을 전송하는 함수입니다.

9개 인자값을 가진 주식주문 함수이며 리턴값이 0이면 성공이며 나머지는 에러입니다.

1초에 5회만 주문가능하며 그 이상 주문요청하면 에러 -308을 리턴합니다.

※ 시장가주문시 주문가격은 0으로 입력합니다. 주문가능수량은 해당 종목의 상한가 기준으로 계산됩니다.

※ 취소주문일때 주문가격은 0으로 입력합니다.

[거래구분]

00 : 지정가

03 : 시장가

05 : 조건부지정가

06 : 최유리지정가

07 : 최우선지정가

10 : 지정가IOC

13 : 시장가IOC

16 : 최유리IOC

20 : 지정가FOK

23 : 시장가FOK

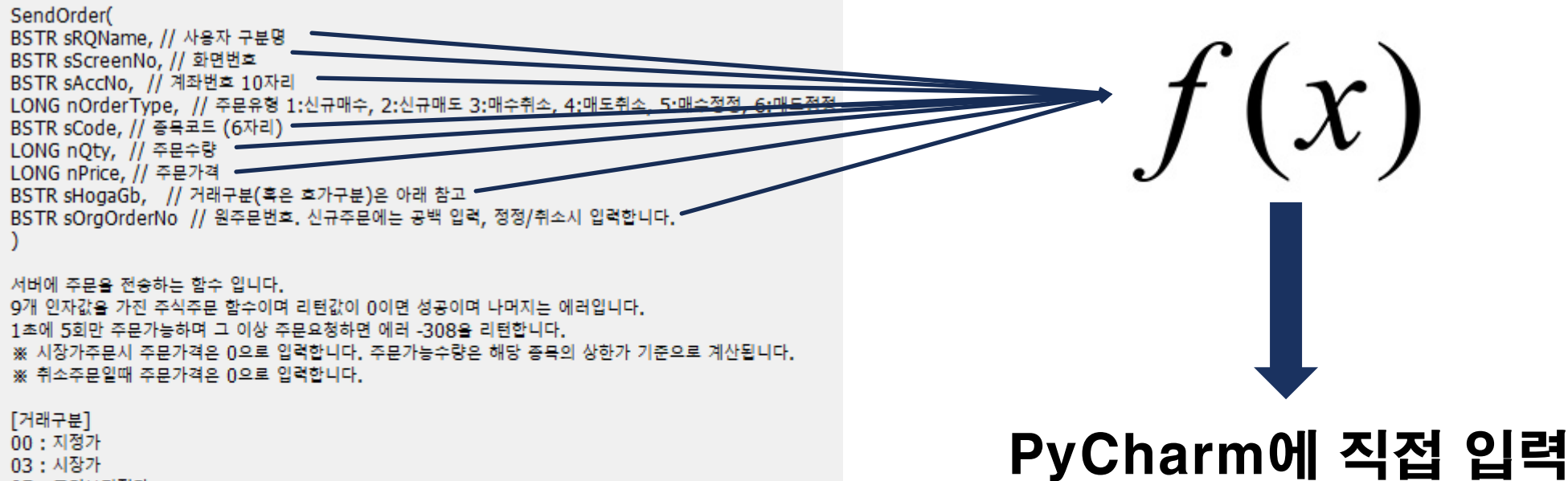
26 : 최유리FOK

61 : 장전시간외증가

62 : 시간외단일가매매

81 : 장후시간외증가

※ 모의투자에서는 지정가 주문과 시장가 주문만 가능합니다.



$f(x)$

PyCharm에 직접 입력

# AUTO TRADING: FUNCTION CALLBACK FROM API

```
self.dynamicCall('SetInputValue(QString, QString)', '종목코드', code)
```

```
self.dynamicCall('SetInputValue(QString, QString)', '수정추가구분', '1')
```

```
if date != None:
```

```
    self.dynamicCall('SetInputValue(QString, QString)', '기준일자', date)
```

```
self.dynamicCall('CommRqData(QString, QString, int, QString)', '주식일봉차트조회', 'opt10081', sPrevNext, self.screen_calculation_stock)
```

# AUTO TRADING: 종목 검색 ALGORITHM

```
code = self.dynamicCall("GetCommData(QString, QString, int, QString)", sTrCode, sRQName, 0, "종목코드")
code = code.strip()
print('%s 일봉데이터 요청' % code)
```

```
cnt = self.dynamicCall('GetRepeatCnt(QString, QString)', sTrCode, sRQName)
print('데이터 일수 %s' % cnt)
```

**최대 600개**

```
for i in range(cnt):
    data = []
```

```
current_price = self.dynamicCall("GetCommData(QString, QString, int, QString)", sTrCode, sRQName, i, "현재가") # 출력 : 000070
value = self.dynamicCall("GetCommData(QString, QString, int, QString)", sTrCode, sRQName, i, "거래량") # 출력 : 000070
trading_value = self.dynamicCall("GetCommData(QString, QString, int, QString)", sTrCode, sRQName, i, "거래대금") # 출력 : 000070
date = self.dynamicCall("GetCommData(QString, QString, int, QString)", sTrCode, sRQName, i, "일자") # 출력 : 000070
start_price = self.dynamicCall("GetCommData(QString, QString, int, QString)", sTrCode, sRQName, i, "시가") # 출력 : 000070
high_price = self.dynamicCall("GetCommData(QString, QString, int, QString)", sTrCode, sRQName, i, "고가") # 출력 : 000070
low_price = self.dynamicCall("GetCommData(QString, QString, int, QString)", sTrCode, sRQName, i, "저가") # 출력 : 000070
```

# AUTO TRADING: 종목 검색 ALGORITHM

```
prev_price = None #과거의 일봉 저가
if bottom_stock_price == True:

    moving_average_price_prev = 0
    price_top_moving = False
    idx = 1
    while True:

        if len(self.calcul_data[idx:]) < 120: #120일치가 있는지 계속 확인
            print('120일치가 없음!')
            break

        total_price = 0
        for value in self.calcul_data[idx:120+idx]:
            total_price += int(value[1])
        moving_average_price_prev = total_price / 120

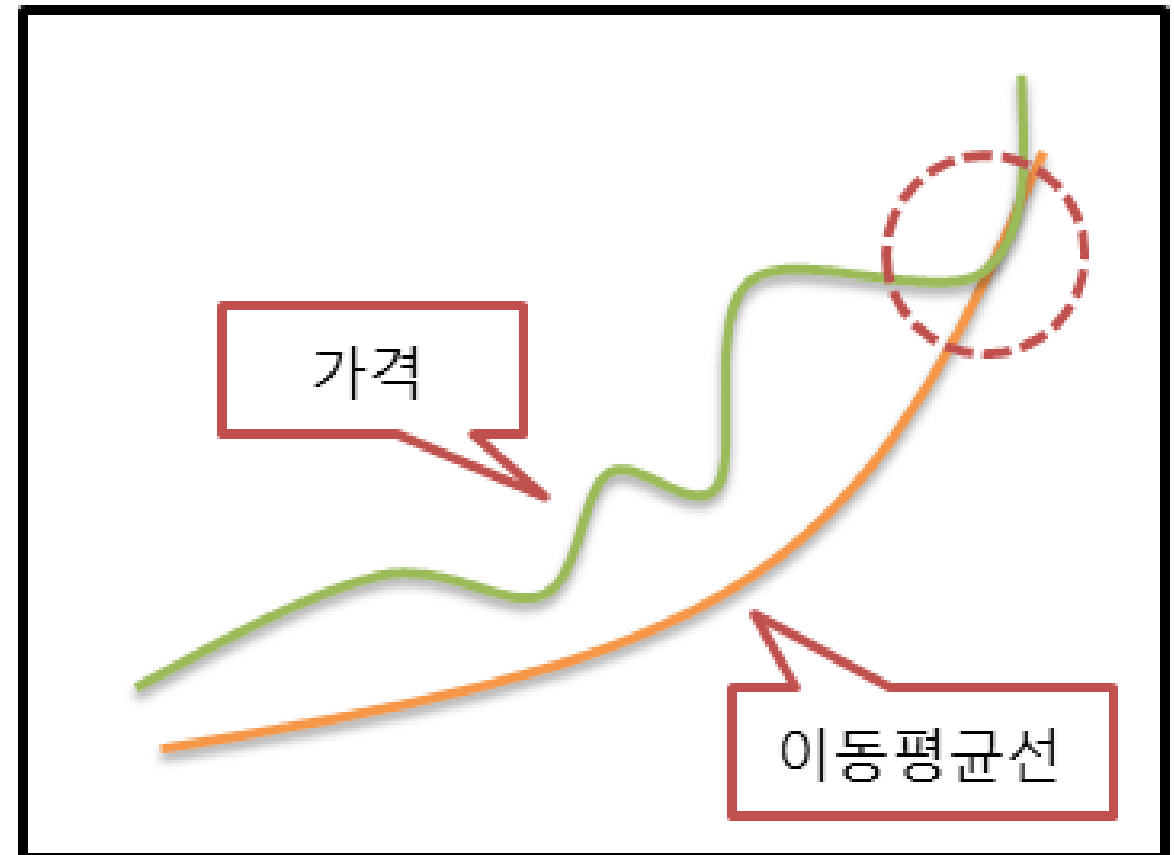
        if moving_average_price_prev <= int(self.calcul_data[idx][6]) and idx <= 5:
            print('5일 동안 주가가 120일 이평선과 같거나 위에 있으면 조건 통과 못함')
            price_top_moving = False
            break

        elif int(self.calcul_data[idx][7]) > moving_average_price_prev and idx > 5:
            print('120일 이평선 위에 있는 일봉 확인됨')
            price_top_moving = True
            prev_price = int(self.calcul_data[idx][7])
            break

        idx += 1

    #해당 부분 이평선이 가장 최근 일자의 이평선 가격보다 낮은지 확인
    if price_top_moving == True:
        if moving_average_price > moving_average_price_prev and check_price > prev_price:
            print('포착된 이평선의 가격이 오늘자(최근일자) 이평선 가격보다 낮은 것 확인됨')
            print('포착된 부분의 일봉 저가가 오늘자 일봉의 고가보다 낮은지 확인됨')
            pass_success = True
```

## 그랜빌 법칙





# AUTO TRADING: 종목 검색 ALGORITHM

```
prev_price = None #과거의 일봉 저가
if bottom_stock_price == True:

    moving_average_price_prev = 0
    price_top_moving = False
    idx = 1
    while True:

        if len(self.calcul_data[idx:]) < 120: #120일치가 있는지 계속 확인
            print('120일치가 없음!')
            break

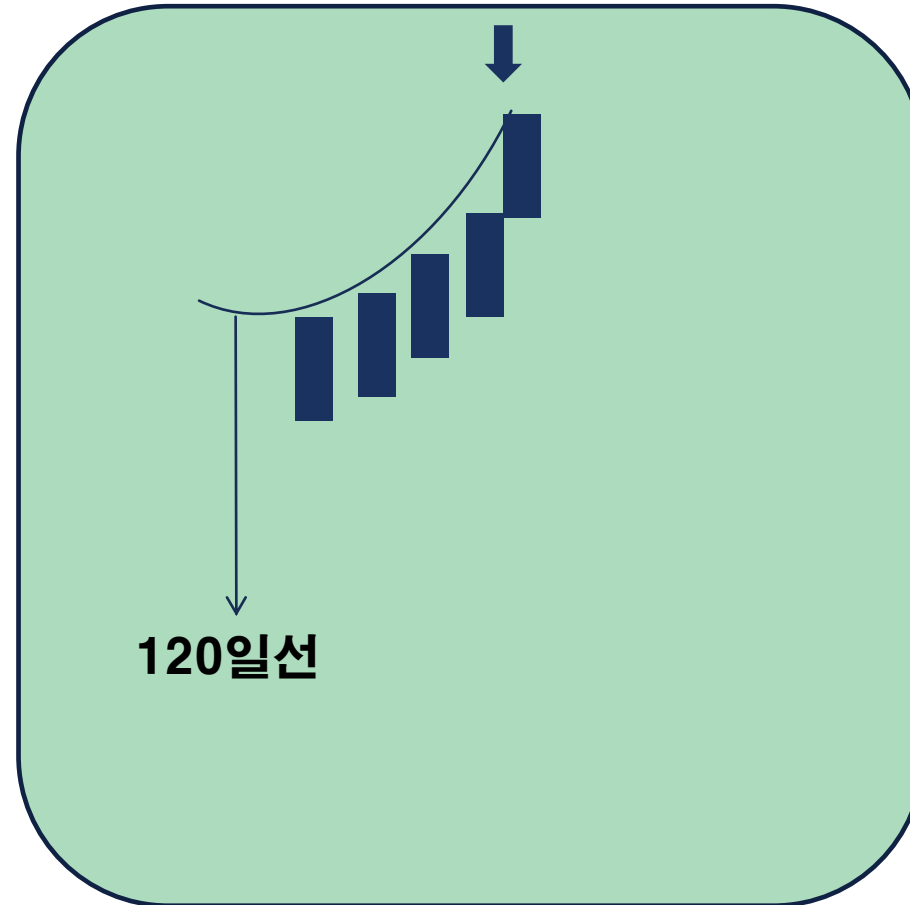
        total_price = 0
        for value in self.calcul_data[idx:120+idx]:
            total_price += int(value[1])
        moving_average_price_prev = total_price / 120

        if moving_average_price_prev <= int(self.calcul_data[idx][6]) and idx <= 5:
            print('5일 동안 주가가 120일 이평선과 같거나 위에 있으면 조건 통과 못함')
            price_top_moving = False
            break

        elif int(self.calcul_data[idx][7]) > moving_average_price_prev and idx > 5:
            print('120일 이평선 위에 있는 일봉 확인됨')
            price_top_moving = True
            prev_price = int(self.calcul_data[idx][7])
            break

        idx += 1

    #해당 부분 이평선이 가장 최근 일자의 이평선 가격보다 낮은지 확인
    if price_top_moving == True:
        if moving_average_price > moving_average_price_prev and check_price > prev_price:
            print('포착된 이평선의 가격이 오늘자(최근일자) 이평선 가격보다 낮은 것 확인됨')
            print('포착된 부분의 일봉 저가가 오늘자 일봉의 고가보다 낮은지 확인됨')
            pass_success = True
```



# AUTO TRADING: 종목 검색 ALGORITHM

```
moving_average_price = total_price / 120

#오늘자 주가가 120일 이평선에 걸쳐있는지 확인
bottom_stock_price = False
check_price = None

if int(self.calcul_data[0][7]) <= moving_average_price and moving_average_price <= int(self.calcul_data[0][6]):
    print('오늘 주가 120이평선에 걸쳐있는 것 확인')
    bottom_stock_price = True
    check_price = int(self.calcul_data[0][6])
```

# AUTO TRADING: 종목 검색 ALGORITHM

```
if bottom_stock_price == True:

    moving_average_price_prev = 0
    price_top_moving = False
    idx = 1
    while True:

        if len(self.calcul_data[idx:]) < 120: #120일치가 있는지 계속 확인
            print('120일치가 없음!')
            break

        total_price = 0
        for value in self.calcul_data[idx:120+idx]:
            total_price += int(value[1])
        moving_average_price_prev = total_price / 120

        if moving_average_price_prev <= int(self.calcul_data[idx][6]) and idx <= 5:
            print('5일 동안 주가가 120일 이평선과 같거나 위에 있으면 조건 통과 못함')
            price_top_moving = False
            break

        elif int(self.calcul_data[idx][7]) > moving_average_price_prev and idx > 5:
            print('120일 이평선 위에 있는 알봉 확인됨')
            price_top_moving = True
            prev_price = int(self.calcul_data[idx][7])
            break

        idx += 1
    #해당 부분 이평선이 가장 최근 일자의 이평선 가격보다 낮은지 확인
    if price_top_moving == True:
        if moving_average_price > moving_average_price_prev and check_price > prev_price:
            print('포착된 이평선의 가격이 오늘자(최근일자) 이평선 가격보다 낮은 것 확인됨')
            print('포착된 부분의 알봉 저가가 오늘자 알봉의 고가보다 낮은지 확인됨')
            pass_success = True
```

종가 } 120일선의 이평선

# AUTO TRADING: 종목 검색 ALGORITHM

```
if pass_success == True:
    print('조건부 통과됨')

    code_nm = self.dynamicCall('GetMasterCodeName(QString)', code)

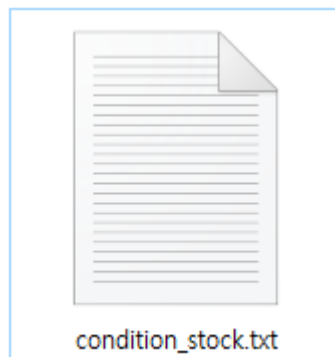
    f = open('C:/Users/bit/PycharmProjects/pythonProject/files/condition_stock.txt', 'a', encoding='utf8')
    f.write('%s\t%s\t%s\n' % (code, code_nm, str(self.calcul_data[0][1])))
    f.close()

elif pass_success == False:
    print('조건부 통과 못함')

self.calcul_data.clear()
self.calculator_event_loop.exit()
```

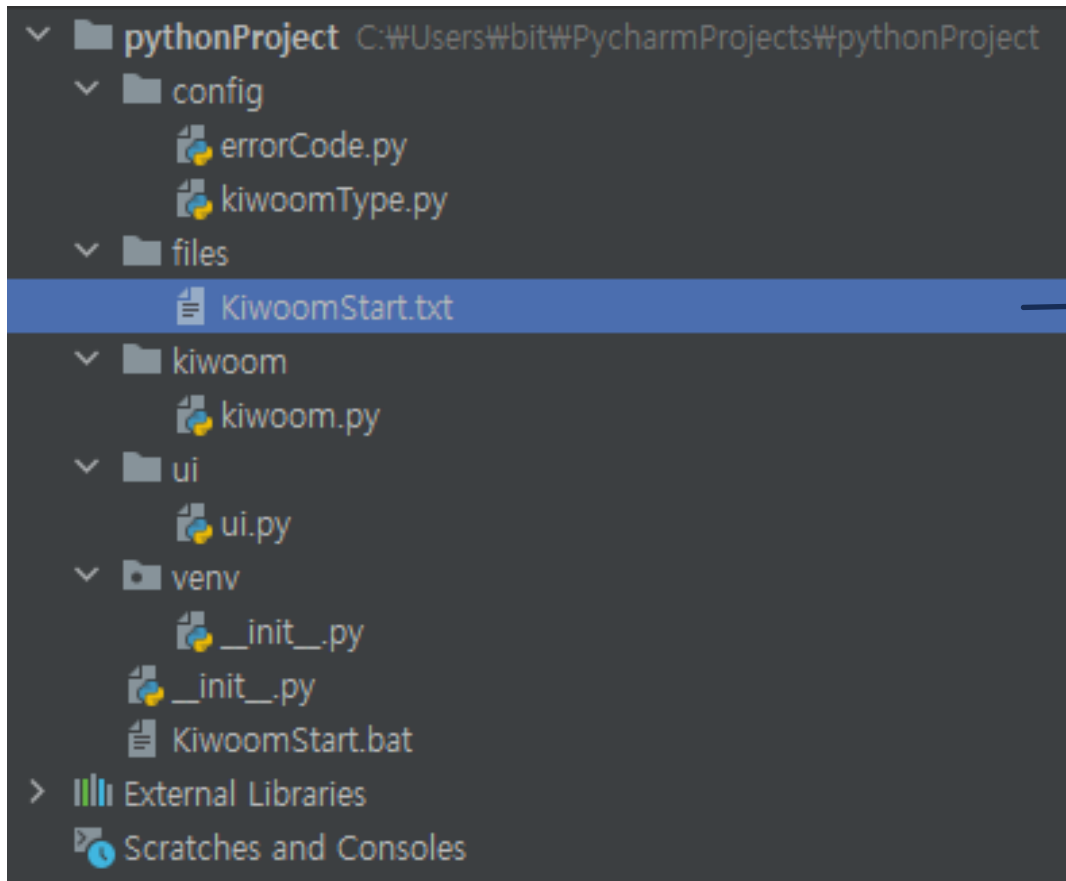
# AUTO TRADING: 종목 저장

종목 검색 중 종목 추출



\*condition\_stock.txt - Windows 메모장  
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)  
041190 우리기술투자 9120

# AUTO TRADING: 종목 저장



저장 확인

# AUTO TRADING: 실시간 거래

```
def realdata_slot(self, sCode, sRealType, sRealData):  
  
    if sRealType == "장시작시간":  
        fid = self.realType.REALTYPE[sRealType]['장운영구분']  
        value = self.dynamicCall("GetCommRealData(QString, int)", sCode, fid)  
  
        if value == '0':  
            print("장 시작 전")  
  
        elif value == '3':  
            print("장 시작")  
  
        elif value == "2":  
            print("장 종료, 동시호가로 넘어감")  
  
        elif value == "4":  
            print("3시 30분 장 종료")  
  
        for code in self.portfolio_stock_dict.keys():  
            self.dynamicCall("SetRealRemove(QString, QString)", self.portfolio_stock_dict[code]['스크린번호'], code)  
  
        QTest.QWait(5000)  
  
        self.file_delete()  
        self.calculator_fnc()
```

# AUTO TRADING: 실시간 거래

```
if sCode in self.account_stock_dict.keys() and sCode not in self.jango_dict.keys():
```

```
    asd = self.account_stock_dict[sCode]
```

```
    meme_rate = (b - asd['매입가']) / asd['매입가'] * 100
```

→ 등락율

```
    if asd['매매가능수량'] > 0 and (meme_rate > 5 or meme_rate < -5):
```

```
        order_success = self.dynamicCall(
```

```
            "SendOrder(QString, QString, QString, int, QString, int, int, QString, QString)",
```

```
            ["신규매도", self.portfolio_stock_dict[sCode]["주문용스크린번호"], self.account_num, 2, sCode,
```

```
            asd['매매가능수량'], 0, self.realType.SENDTYPE['거래구분']['시장가'], ""]
```

```
        )
```

```
    if order_success == 0:
```

```
        print("매도주문 전달 성공")
```

```
        del self.account_stock_dict[sCode]
```

```
    else:
```

```
        print("매도주문 전달 실패")
```



# AUTO TRADING: 실시간 거래

1 [0343] 기간별 주문체결상세(모의투자)

|      |           |               |      |      |      |  |   |            |         |     |
|------|-----------|---------------|------|------|------|--|---|------------|---------|-----|
| 계좌번호 | 8018-6056 | 모의_상시_metainv | 비밀번호 | **** | 주문일자 | 2022/03/21   | ~ | 2022/04/05 | 당일      | 1개월 |
| 주식채권 | 주식        |               | 시장구분 | 전체   | 조회구분 | <input checked="" type="radio"/> 전체 <input type="radio"/> 체결 |   |            | 조회      | 다음  |
| 매매구분 | 전체        |               | 종목코드 |      | 약정금액 |  | 0 |            | 매체별약정현황 |     |

| 주식채권       | 주문번호   | 원주문번호  | 종목번호   | 매매구분 | 주문유형구분 | 주문수량  | 주문단가  | 확인수량  | 체결번호     |
|------------|--------|--------|--------|------|--------|-------|-------|-------|----------|
| 주문일자       | 종목명    | 접수구분   | 신용거래구분 | 체결수량 | 체결평균단가 | 정정/취소 | 통신    | 예약/반대 | 체결시간     |
| 주식         | 7166   |        | 041190 | 지정가  | 현금매수   | 131   | 9,350 | 131   |          |
| 2022/03/29 | 우리기술투자 | 접수     |        | 131  | 9,350  |       | 오픈API | 일반    | 09:00:01 |
| 주식         | 159539 | 158973 | 189300 | 지정가  | 현금매수   | 3     | 0     | 3     |          |
| 2022/03/28 | 인텔리안테크 | 접수     |        |      |        | 취소    | 영웅문4  | 일반    |          |

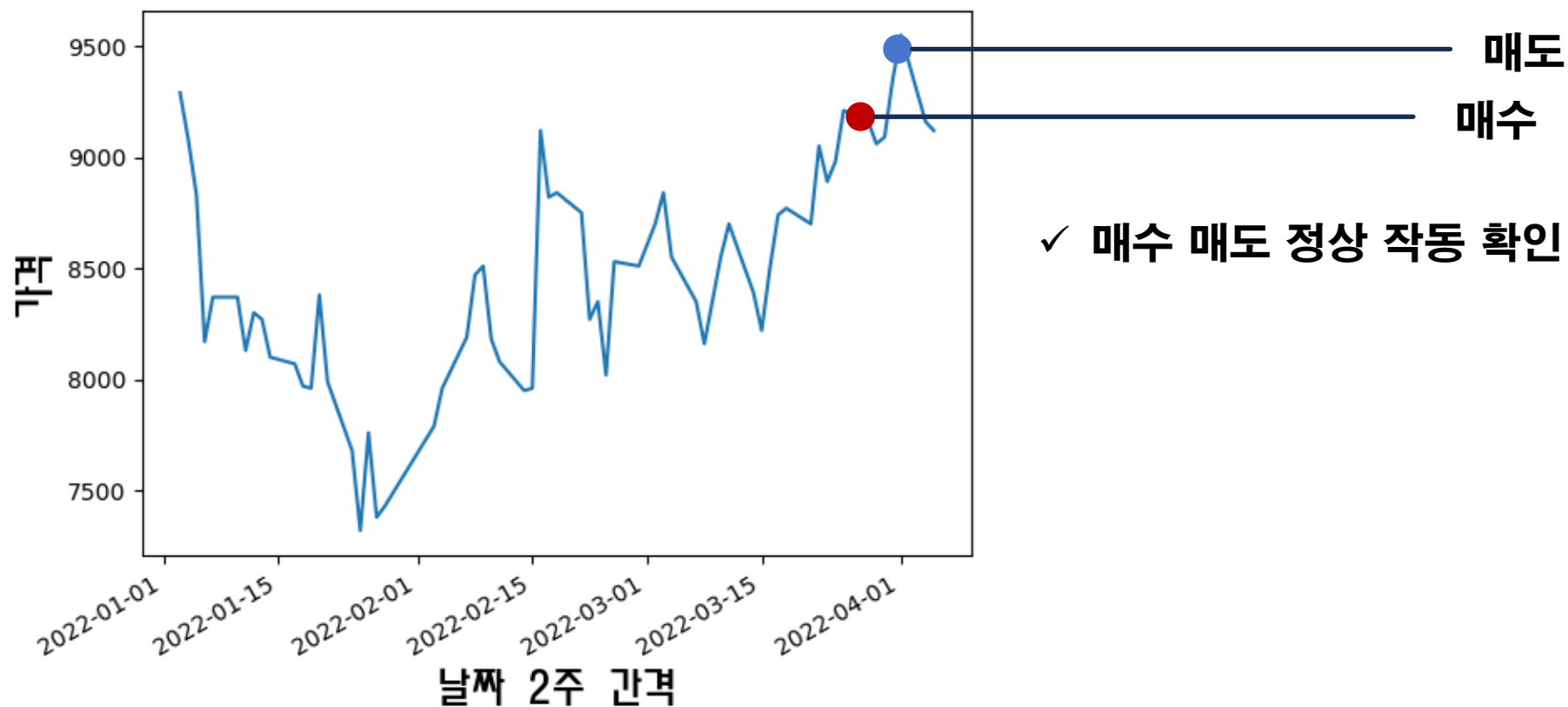
1 [0343] 기간별 주문체결상세(모의투자)

|      |           |               |      |      |      |  |   |            |         |     |
|------|-----------|---------------|------|------|------|--|---|------------|---------|-----|
| 계좌번호 | 8018-6056 | 모의_상시_metainv | 비밀번호 | **** | 주문일자 | 2022/03/21   | ~ | 2022/04/05 | 당일      | 1개월 |
| 주식채권 | 주식        |               | 시장구분 | 전체   | 조회구분 | <input checked="" type="radio"/> 전체 <input type="radio"/> 체결 |   |            | 조회      | 다음  |
| 매매구분 | 전체        |               | 종목코드 |      | 약정금액 |  | 0 |            | 매체별약정현황 |     |

| 주식채권       | 주문번호   | 원주문번호 | 종목번호   | 매매구분 | 주문유형구분  | 주문수량  | 주문단가    | 확인수량  | 체결번호     |
|------------|--------|-------|--------|------|---------|-------|---------|-------|----------|
| 주문일자       | 종목명    | 접수구분  | 신용거래구분 | 체결수량 | 체결평균단가  | 정정/취소 | 통신      | 예약/반대 | 체결시간     |
| 주식         | 154505 |       | 041190 | 시장가  | 현금매도    | 131   | 0       | 131   |          |
| 2022/04/01 | 우리기술투자 | 접수    |        | 131  | 9,800   |       | 오픈API   | 일반    | 14:10:02 |
| 주식         | 147687 |       | 066570 | 지정가  | 현금매수    | 10    | 121,000 | 10    |          |
| 2022/03/30 | L&G전자  | 접수    |        | 10   | 121,000 |       | 영웅문4    | 일반    | 14:23:23 |

# AUTO TRADING: 실시간 거래

<우리기술투자 주식 가격>



# AUTO TRADING: 실시간 거래

1 [0343] 기간별 주문체결상세 (모의투자)

계좌번호 8018-6056 모의\_상시\_meta.inve 비밀번호 \*\*\*\* 주문일자 2022/03/21 ~ 2022/04/05 당일 1개월  
 주식채권 주식 시장구분 전체 조회구분 ☒ 전체 ☐ 체결 조회 다음  
 매매구분 전체 종목코드 약정금액 0 매체별약정현황

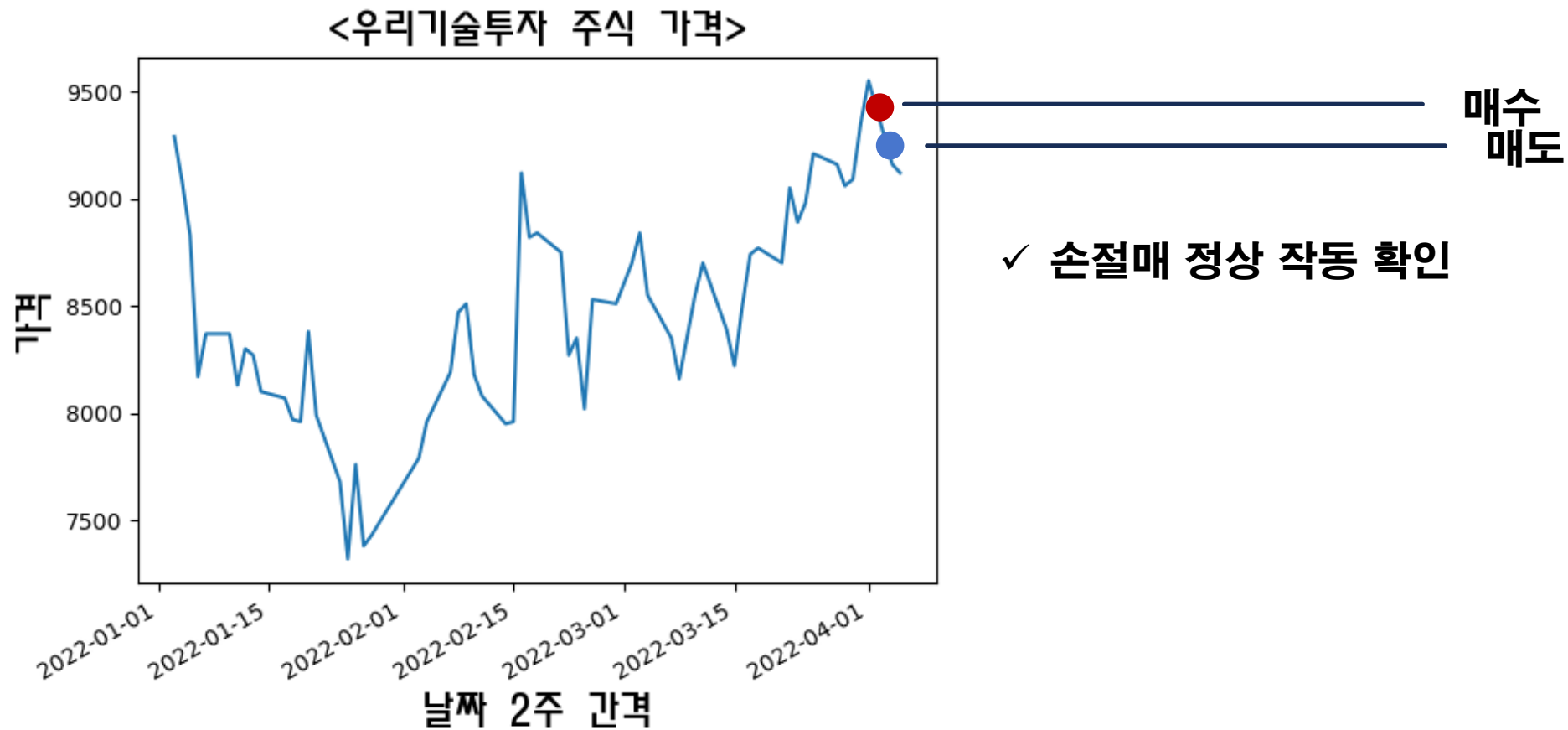
| 주식채권       | 주문번호   | 원주문번호 | 종목번호   | 매매구분 | 주문유형구분 | 주문수량  | 주문단가  | 확인수량  | 체결번호     |
|------------|--------|-------|--------|------|--------|-------|-------|-------|----------|
| 주문일자       | 종목명    | 접수구분  | 신용거래구분 | 체결수량 | 체결평균단가 | 정정/취소 | 통신    | 예약/반대 | 체결시간     |
| 주식         | 154506 |       | 041190 | 지정가  | 현금매수   | 117   | 9,770 | 117   |          |
| 2022/04/01 | 우리기술투자 | 접수    |        | 117  | 9,770  |       | 오픈API | 일반    | 14:10:02 |

1 [0343] 기간별 주문체결상세 (모의투자)

계좌번호 8018-6056 모의\_상시\_meta.inve 비밀번호 \*\*\*\* 주문일자 2022/03/21 ~ 2022/04/05 당일 1개월  
 주식채권 주식 시장구분 전체 조회구분 ☒ 전체 ☐ 체결 조회 다음  
 매매구분 전체 종목코드 약정금액 0 매체별약정현황

| 주식채권       | 주문번호   | 원주문번호 | 종목번호   | 매매구분 | 주문유형구분 | 주문수량  | 주문단가  | 확인수량  | 체결번호     |
|------------|--------|-------|--------|------|--------|-------|-------|-------|----------|
| 주문일자       | 종목명    | 접수구분  | 신용거래구분 | 체결수량 | 체결평균단가 | 정정/취소 | 통신    | 예약/반대 | 체결시간     |
| 주식         | 19691  |       | 041190 | 시장가  | 현금매도   | 117   | 0     | 117   |          |
| 2022/04/05 | 우리기술투자 | 접수    |        | 117  | 9,140  |       | 오픈API | 일반    | 09:08:02 |

# AUTO TRADING: 실시간 거래



# AUTO TRADING: 실시간 거래 종료

```
def realdata_slot(self, sCode, sRealType, sRealData):  
  
    if sRealType == "장시작시간":  
        fid = self.realType.REALTYPE[sRealType]['장운영구분']  
        value = self.dynamicCall("GetCommRealData(QString, int)", sCode, fid)  
  
        if value == '0':  
            print("장 시작 전")  
  
        elif value == '3':  
            print("장 시작")  
  
        elif value == "2":  
            print("장 종료, 동시호가로 넘어감")  
  
        elif value == "4":  
            print("3시 30분 장 종료")  
  
            for code in self.portfolio_stock_dict.keys():  
                self.dynamicCall("SetRealRemove(QString, QString)", self.portfolio_stock_dict[code]['스크린번호'], code)  
  
            QTest.qWait(5000)  
  
            self.file_delete()  
            self.calculator_fnc()
```

# AUTO TRADING: 실시간 거래 종료

Kiwoom Start

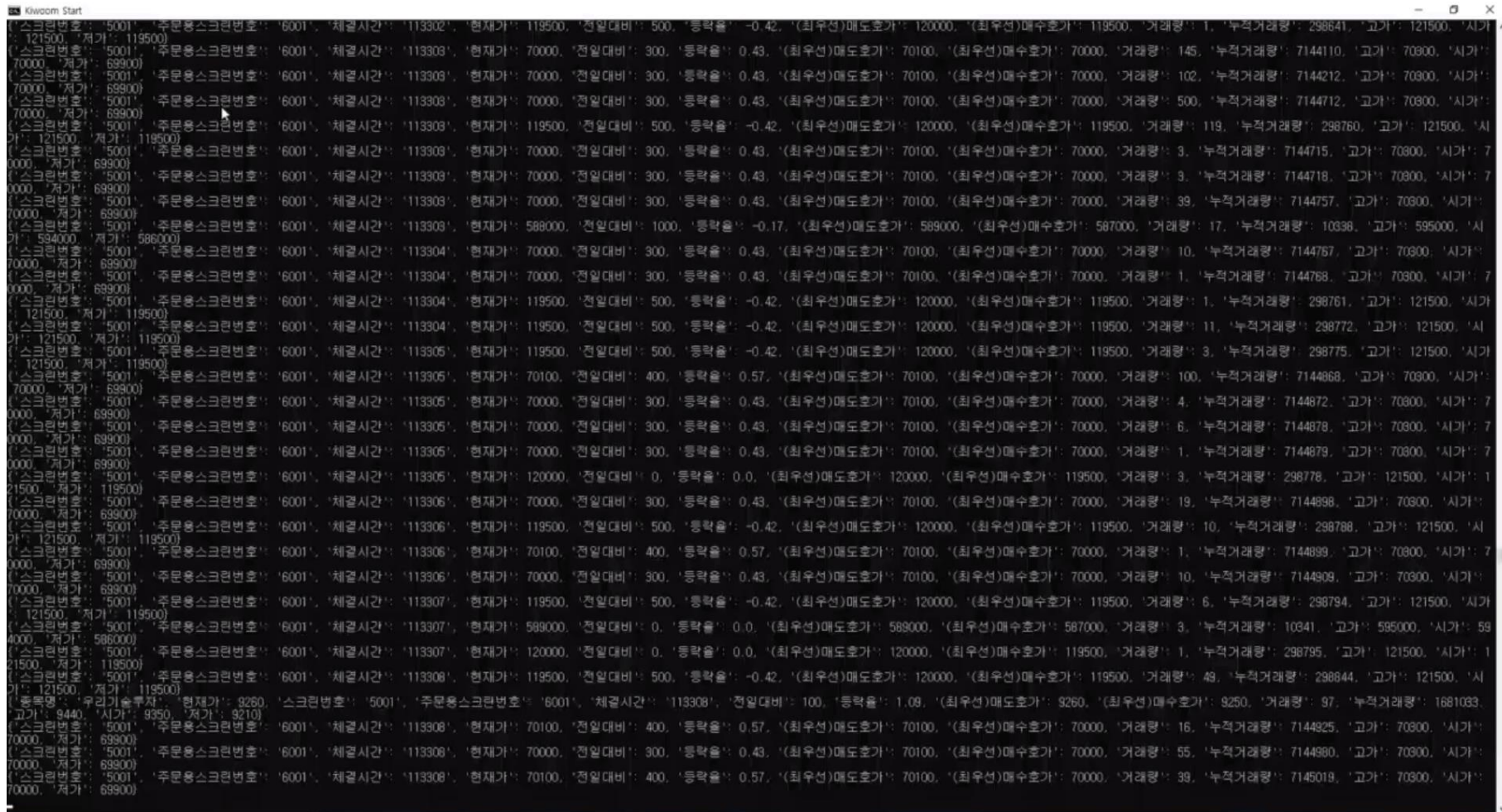
118500, '저가': 117000}  
{ '스크린번호': '5001', '주문용스크린번호': '6001', '체결시간': '151959', '현재가': 117500, '전일대비': 500, '등락율': 0.43, '(최우선)매도호가': 117500, '(최우선)매수호가': 117000, '거래량': 10, '누적거래량': 599170, '고가': 119000, '시가': 118500, '저가': 117000}  
{ '스크린번호': '5001', '주문용스크린번호': '6001', '체결시간': '151959', '현재가': 117500, '전일대비': 500, '등락율': 0.43, '(최우선)매도호가': 117500, '(최우선)매수호가': 117000, '거래량': 1, '누적거래량': 599171, '고가': 119000, '시가': 118500, '저가': 117000}  
{ '스크린번호': '5001', '주문용스크린번호': '6001', '체결시간': '151959', '현재가': 69300, '전일대비': 0, '등락율': 0.0, '(최우선)매도호가': 69300, '(최우선)매수호가': 69200, '거래량': 2, '누적거래량': 8002034, '고가': 69600, '시가': 69400, '저가': 69100}  
{ '스크린번호': '5001', '주문용스크린번호': '6001', '체결시간': '151959', '현재가': 69300, '전일대비': 0, '등락율': 0.0, '(최우선)매도호가': 69300, '(최우선)매수호가': 69200, '거래량': 116, '누적거래량': 8002150, '고가': 69600, '시가': 69400, '저가': 69100}  
{ '스크린번호': '5001', '주문용스크린번호': '6001', '체결시간': '151959', '현재가': 117000, '전일대비': 0, '등락율': 0.0, '(최우선)매도호가': 117500, '(최우선)매수호가': 117000, '거래량': 10, '누적거래량': 599181, '고가': 119000, '시가': 118500, '저가': 117000}

종료

종목 검색 준비중

3시 30분 장 종료  
코스닥 갱신 1557  
1 / 1557: KOSDAQ Stock Code : 900110 is updating....

# AUTO TRADING: 실시간 거래 화면



The screenshot displays a window titled 'Kiwoom Start' containing a list of auto-trading orders. Each row represents an order with the following fields: '스크린번호' (Screen Number), '주문용스크린번호' (Order Screen Number), '체결시간' (Execution Time), '현재가' (Current Price), '전일대비' (Change from Previous Day), '등락율' (Change Rate), '(최우선)매도호가' (Best Bid), '(최우선)매수호가' (Best Ask), '거래량' (Volume), '누적거래량' (Cumulative Volume), '고가' (High), and '시가' (Open). The orders are listed in a table format, showing various parameters for each trade.

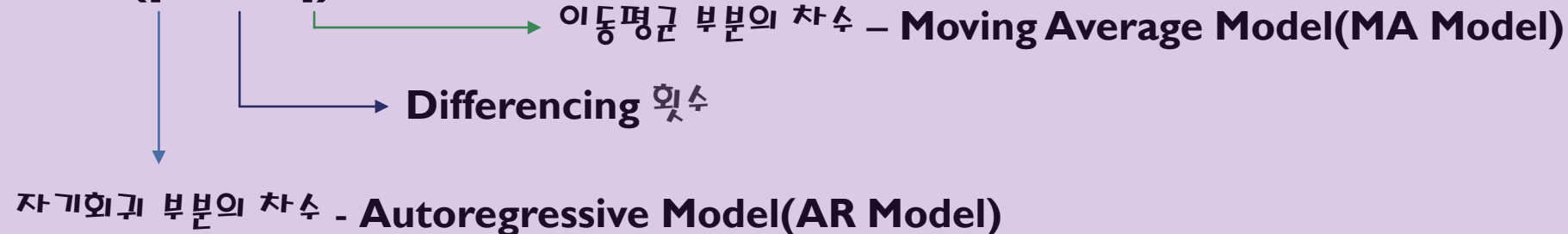
| 스크린번호 | 주문용스크린번호 | 체결시간   | 현재가    | 전일대비 | 등락율   | (최우선)매도호가 | (최우선)매수호가 | 거래량 | 누적거래량   | 고가     | 시가     |
|-------|----------|--------|--------|------|-------|-----------|-----------|-----|---------|--------|--------|
| 5001  | 6001     | 113302 | 119500 | 500  | -0.42 | 120000    | 119500    | 1   | 298841  | 121500 | 121500 |
| 5001  | 6001     | 113308 | 70000  | 300  | 0.43  | 70100     | 70000     | 145 | 7144110 | 70300  | 70300  |
| 5001  | 6001     | 113309 | 70000  | 300  | 0.43  | 70100     | 70000     | 102 | 7144212 | 70300  | 70300  |
| 5001  | 6001     | 113303 | 70000  | 300  | 0.43  | 70100     | 70000     | 500 | 7144712 | 70300  | 70300  |
| 5001  | 6001     | 113303 | 119500 | 500  | -0.42 | 120000    | 119500    | 119 | 298760  | 121500 | 121500 |
| 5001  | 6001     | 113303 | 70000  | 300  | 0.43  | 70100     | 70000     | 3   | 7144715 | 70300  | 70300  |
| 5001  | 6001     | 113303 | 70000  | 300  | 0.43  | 70100     | 70000     | 39  | 7144757 | 70300  | 70300  |
| 5001  | 6001     | 113303 | 589000 | 1000 | -0.17 | 589000    | 587000    | 17  | 10338   | 595000 | 595000 |
| 5001  | 6001     | 113304 | 70000  | 300  | 0.43  | 70100     | 70000     | 10  | 7144767 | 70300  | 70300  |
| 5001  | 6001     | 113304 | 119500 | 500  | -0.42 | 120000    | 119500    | 1   | 298761  | 121500 | 121500 |
| 5001  | 6001     | 113304 | 119500 | 500  | -0.42 | 120000    | 119500    | 11  | 298772  | 121500 | 121500 |
| 5001  | 6001     | 113305 | 70100  | 400  | 0.57  | 70100     | 70000     | 100 | 7144868 | 70300  | 70300  |
| 5001  | 6001     | 113305 | 70000  | 300  | 0.43  | 70100     | 70000     | 4   | 7144872 | 70300  | 70300  |
| 5001  | 6001     | 113305 | 70000  | 300  | 0.43  | 70100     | 70000     | 6   | 7144878 | 70300  | 70300  |
| 5001  | 6001     | 113305 | 70000  | 300  | 0.43  | 70100     | 70000     | 1   | 7144879 | 70300  | 70300  |
| 5001  | 6001     | 113305 | 120000 | 0    | 0.0   | 120000    | 119500    | 3   | 298778  | 121500 | 121500 |
| 5001  | 6001     | 113306 | 70000  | 300  | 0.43  | 70100     | 70000     | 19  | 7144898 | 70300  | 70300  |
| 5001  | 6001     | 113306 | 119500 | 500  | -0.42 | 120000    | 119500    | 10  | 298788  | 121500 | 121500 |
| 5001  | 6001     | 113306 | 70100  | 400  | 0.57  | 70100     | 70000     | 1   | 7144899 | 70300  | 70300  |
| 5001  | 6001     | 113306 | 70000  | 300  | 0.43  | 70100     | 70000     | 10  | 7144909 | 70300  | 70300  |
| 5001  | 6001     | 113307 | 119500 | 500  | -0.42 | 120000    | 119500    | 6   | 298794  | 121500 | 121500 |
| 5001  | 6001     | 113307 | 589000 | 0    | 0.0   | 589000    | 587000    | 3   | 10341   | 595000 | 595000 |
| 5001  | 6001     | 113307 | 120000 | 0    | 0.0   | 120000    | 119500    | 1   | 298795  | 121500 | 121500 |
| 5001  | 6001     | 113308 | 119500 | 500  | -0.42 | 120000    | 119500    | 49  | 298844  | 121500 | 121500 |
| 5001  | 6001     | 113308 | 70100  | 400  | 0.57  | 70100     | 70000     | 39  | 7145019 | 70300  | 70300  |

# ARIMA

## ARIMA 모델

- **시계열 데이터의 정상성 (Stationary)**  
시간에 관계없이 평균과 분산이 일정한 시계열 데이터 → ACP패턴을 보고 추측가능
- **시계열 데이터의 비정상성 (Nonstationary)**  
시간에 따라 분산이나 평균이 일정하지 않은 경우
- **시계열의 차분(differencing)**  
현 시점 데이터에서 d시점 이전 데이터를 뺀 것

**ARIMA(p, d, q)**





# ARIMA

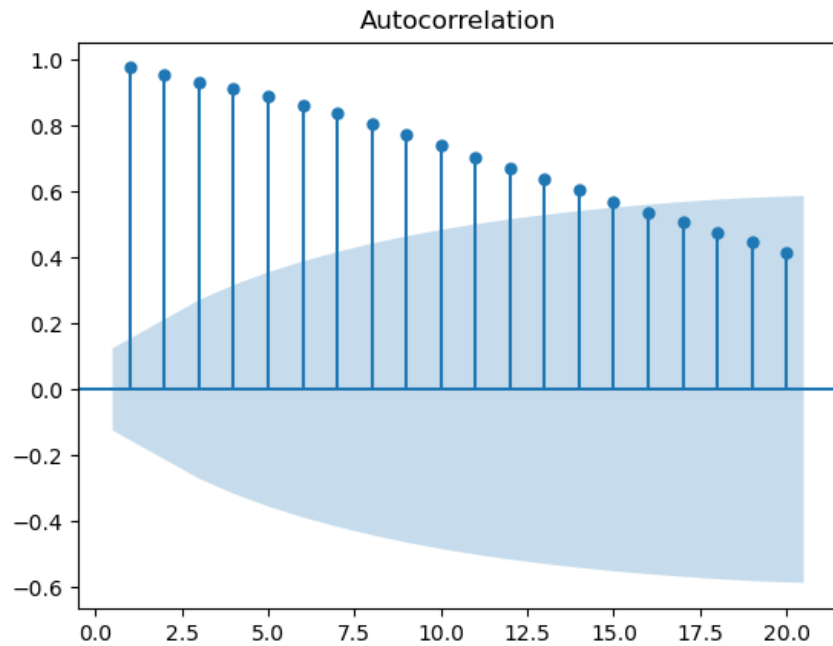
우리기술투자( '041190' )

2021년~현재까지 종가 그래프

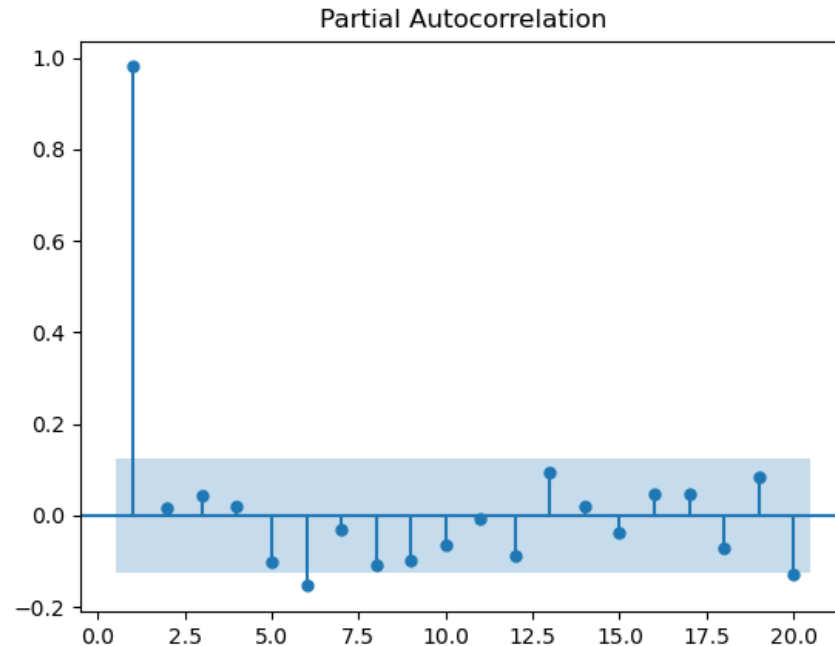


# ARIMA

## ARIMA 모델 - Correlation Function



ACP(Auto Correlation Function)



PACP(Partial Auto Correlation Function)

- ✓ 데이터가 Nonstationary한 형태
- ✓ 차분(differencing) 해주어야 함

# ARIMA

ARIMA 모델 - Best Model 찾기

```
Performing stepwise search to minimize aic
ARIMA(2,1,2)(0,0,0)[0] intercept : AIC=inf, Time=0.26 sec
ARIMA(0,1,0)(0,0,0)[0] intercept : AIC=2577.444, Time=0.01 sec
ARIMA(1,1,0)(0,0,0)[0] intercept : AIC=2578.486, Time=0.01 sec
ARIMA(0,1,1)(0,0,0)[0] intercept : AIC=2578.479, Time=0.01 sec
ARIMA(0,1,0)(0,0,0)[0] : AIC=2575.704, Time=0.00 sec
ARIMA(1,1,1)(0,0,0)[0] intercept : AIC=2580.473, Time=0.07 sec

Best model: ARIMA(0,1,0)(0,0,0)[0]
Total fit time: 0.373 seconds
```

ARIMA(0, 1, 0)의 형태는 **보행모형**  
예측치들이 마지막 관측치가 되는 모델  
But, Best Model이니 일단 써보자!

AIC가 가장 작은 모델로 채택

# ARIMA

## ARIMA 모델 – Summary()

```
SARIMAX Results
=====
Dep. Variable:          y      No. Observations:          173
Model:                 SARIMAX(0, 1, 0)      Log Likelihood      -1286.852
Date:                 Tue, 05 Apr 2022      AIC                  2575.704
Time:                 00:47:59      BIC                  2578.851
Sample:              0      HQIC                  2576.981
                   - 173
Covariance Type:      opg
=====
              coef      std err          z      P>|z|      [0.025      0.975]
-----
sigma2      1.843e+05   9929.030     18.563     0.000     1.65e+05     2.04e+05
=====
Ljung-Box (L1) (Q):           0.83      Jarque-Bera (JB):          280.25
Prob(Q):                     0.36      Prob(JB):              0.00
Heteroskedasticity (H):       0.68      Skew:                  1.12
Prob(H) (two-sided):         0.16      Kurtosis:              8.84
=====
```

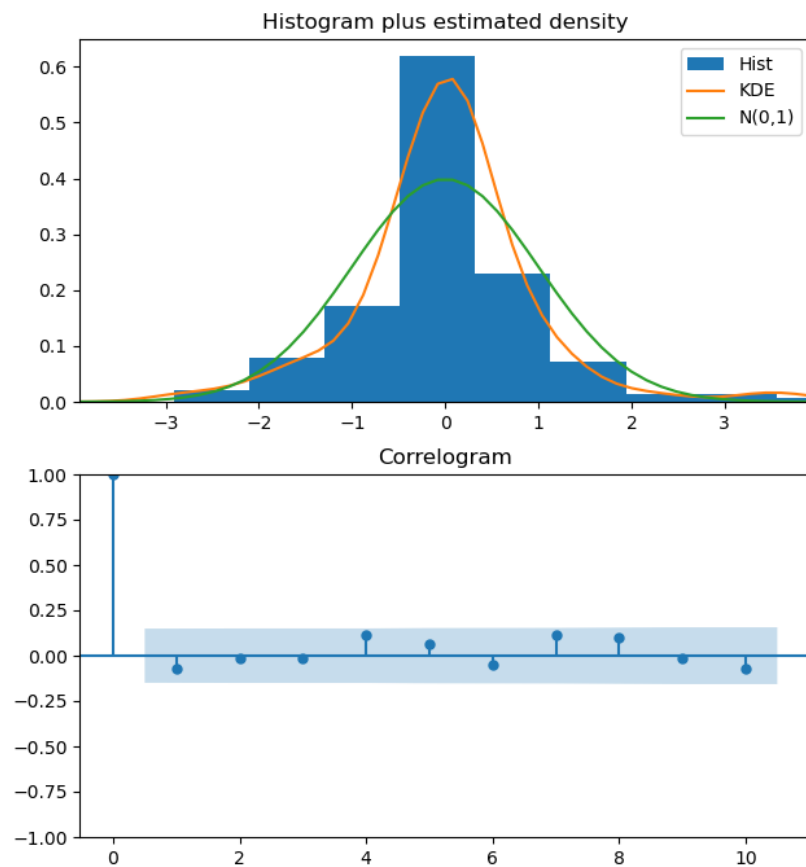
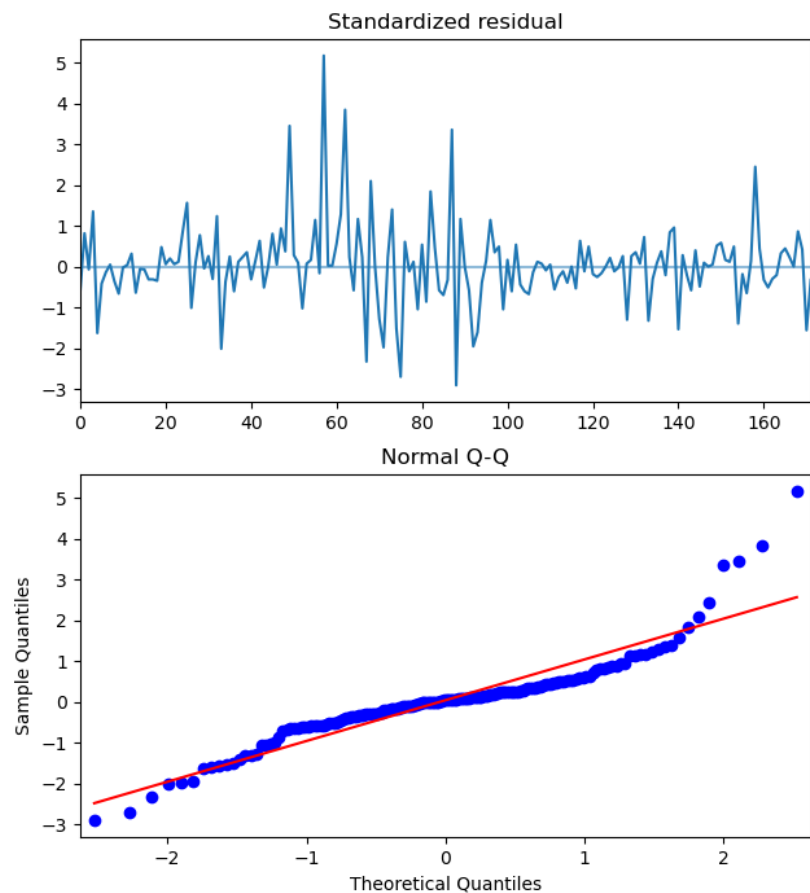
❖ 귀무가설 “잔차(residual)가 백색잡음(white noise) 시계열을 따른다”

유의수준 0.05에서 귀무가설을 기각하지 못함(Prob (Q) = 0.65)

→ 시계열 모형이 잘 적합되었으며, 남은 잔차는 더이상 자기상관을 가지지 않는 백색 잡음임

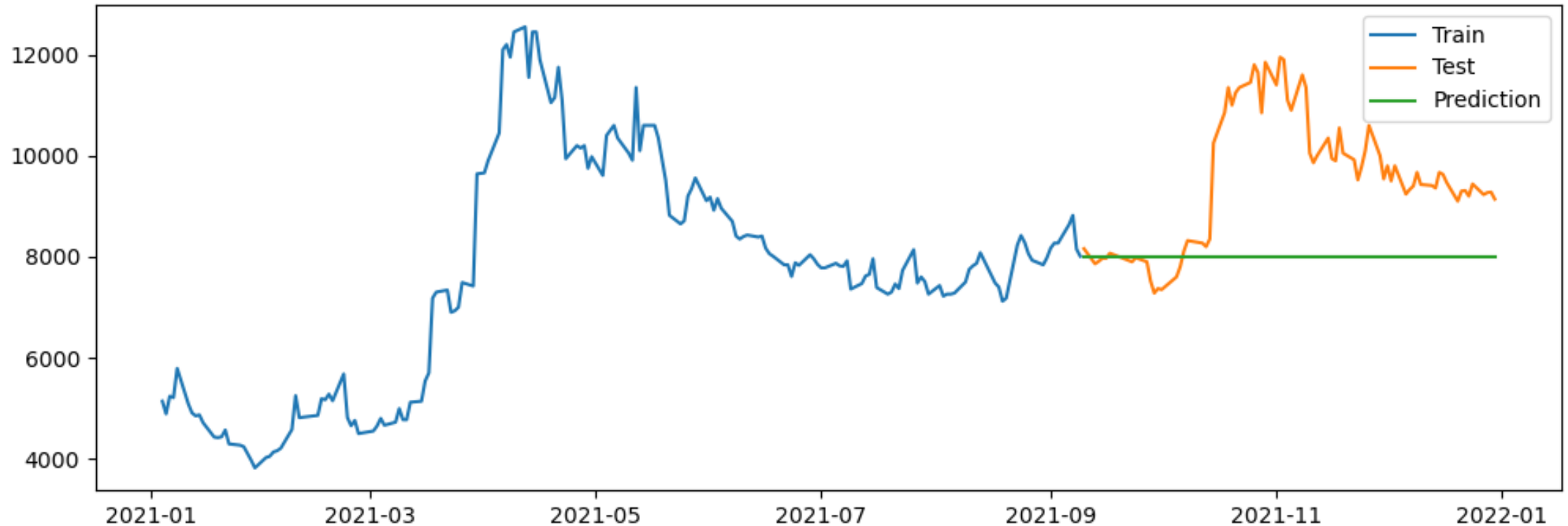
# ARIMA

## ARIMA 모델 잔차 검토 시각화



# ARIMA

## ARIMA 모델 – 예측 시각화



**한 스텝씩 예측하여 데이터를 관측할때마다 모형을 업데이트하는 방식적용!!**

# ARIMA

## ARIMA 모델 Update

```
SARIMAX Results
=====
Dep. Variable:          y      No. Observations:          248
Model:                SARIMAX(0, 1, 0)  Log Likelihood      -1847.399
Date:                Tue, 05 Apr 2022    AIC                3696.798
Time:                00:49:08    BIC                3700.307
Sample:              0      HQIC                3698.211
                  - 248
Covariance Type:      opg
=====
              coef    std err          z      P>|z|      [0.025    0.975]
-----
sigma2      1.84e+05   8505.102    21.639    0.000    1.67e+05    2.01e+05
=====
Ljung-Box (L1) (Q):      0.71    Jarque-Bera (JB):      351.21
Prob(Q):                0.40    Prob(JB):           0.00
Heteroskedasticity (H):  0.70    Skew:               1.02
Prob(H) (two-sided):    0.11    Kurtosis:           8.48
=====
```

데이터가 늘어남

다시 계산되어 update됨

이전 모델보다 AIC 가 늘어남  
Update한 모델의 성능이 더 나빠진 것 확인

# ARIMA

## ARIMA 모델 Update 시각화

ARIMA(0,1,0)모형



**결론 : 원하는 방식으로 ARIMA 모델 Update는 성공적으로 이루어 졌으나,  
데이터 특성 상 ARIMA모델로 정확한 가격을 예측하는 것이 어려움**



# 한계점: API + ALGORITHM

```
def day_kiwoom_db(self, code=None, date=None, sPrevNext='0'):  
    QTest.qWait(3600)
```

: 긴 시간 소요

한종목당 전체 일봉 개수 중 600개씩 끊어서  
3.6초당 분석

# 한계점: ARIMA

## ARIMA



**데이터 특성 상 ARIMA모델로 정확한 가격을 예측하는 것이 어려움**

# MACHINE LEARNING

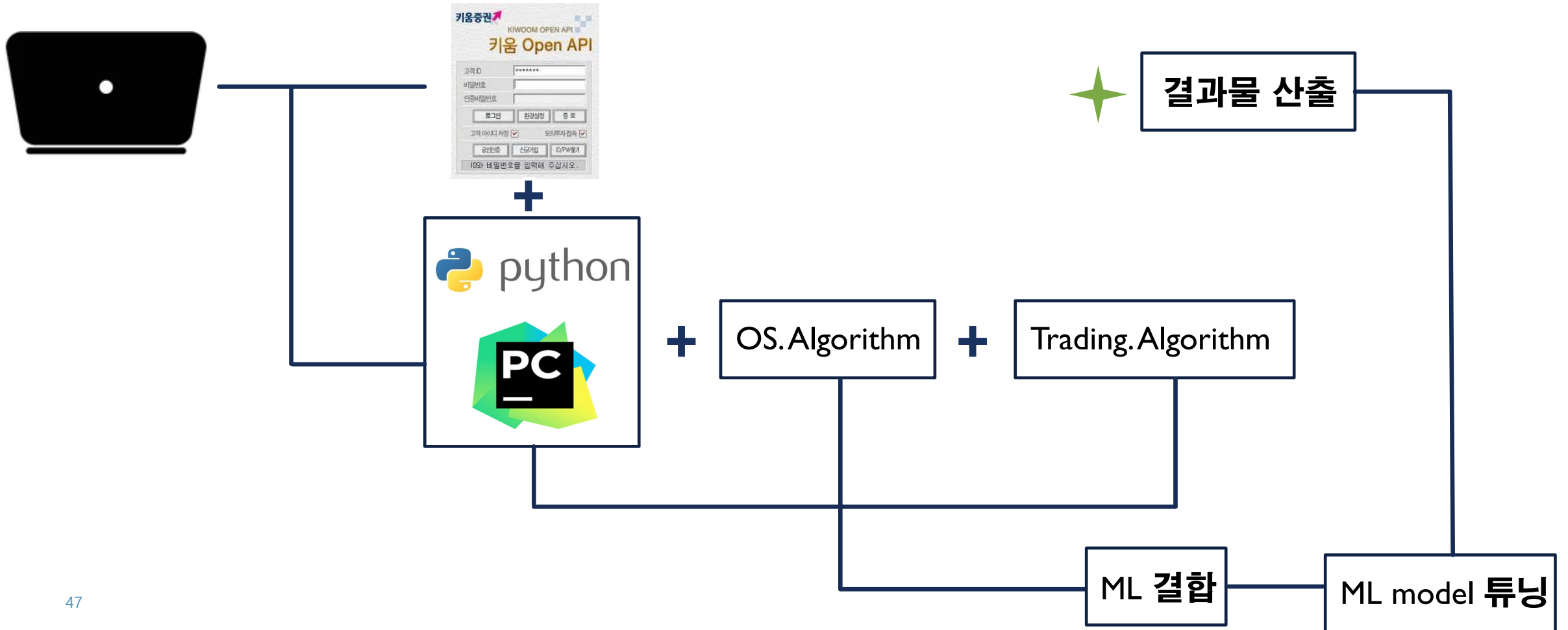
1. 로그인 : 작업 스케줄러 Application과 + OPEN API 를 이용한 자동 로그인
2. 종목검색 : KIWOOM OPEN API와 PY QT BASE ALGORITHM을 이용한 종목 검색
3. 실시간 거래: KIWOOM OPEN API와 PY QT BASE ALGORITHM을 이용한 종목 검색
4. 장종료: 다시 종목 검색으로 2번으로 돌아가 다시 동일 작업 시행

# MACHINE LEARNING

## Machine Learning

1. 2년간의 Dataset을 이용하여 Back Staking 한 후 제일 유사한 가중치를 저장
2. 전일 기준 Dataset을 통해 Machine Learning Model에 가중치를 Load
3. 학습 후 다음날 **상승 예상 종목** 5개를 추출
4. 매매 알고리즘을 통하여 5개 종목을 focusing 하여 매매 하는 전략

# DEVELOPING PROCESS



# TIME LINE

|                  | 2월 3째주 | 2월 4째주 | 3월 1째주 | 3월 2째주 | 3월 3째주 | 3월 4째주 | 3월 5째주 | 4월 1째주 | 4월 2째주 | 4월 3째주 | 4월 4째주 | 5월 1째주 | 5월 2째주 |
|------------------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|
| 개발 가상환경설정        |        |        |        |        |        |        |        |        |        |        |        |        |        |
| 키움증권 계좌 개설       |        |        |        |        |        |        |        |        |        |        |        |        |        |
| API + 계좌 연동      |        |        |        |        |        |        |        |        |        |        |        |        |        |
| DATASET 확보 및 전처리 |        |        |        |        |        |        |        |        |        |        |        |        |        |
| PC 자동화 시스템 설정    |        |        |        |        |        |        |        |        |        |        |        |        |        |
| 매매 알고리즘 설정       |        |        |        |        |        |        |        |        |        |        |        |        |        |
| ML 모델 설정         |        |        |        |        |        |        |        |        |        |        |        |        |        |
| ML 모델링 튜닝        |        |        |        |        |        |        |        |        |        |        |        |        |        |
| ML 결합 및 구동       |        |        |        |        |        |        |        |        |        |        |        |        |        |
| 시각화 및 모델개선       |        |        |        |        |        |        |        |        |        |        |        |        |        |
| 결과물 발표 준비        |        |        |        |        |        |        |        |        |        |        |        |        |        |



Q&A