# 이베스트증권 XingAPI

# Python 을 이용한  API 자동매매 개발 가이드

## 초보자용(For Beginners)

## 2020년 7월 수정본


## 제작 및 배포 : 파생인의쉼터 말과의미

(http://cafe.naver.com/fo24)


# 목차

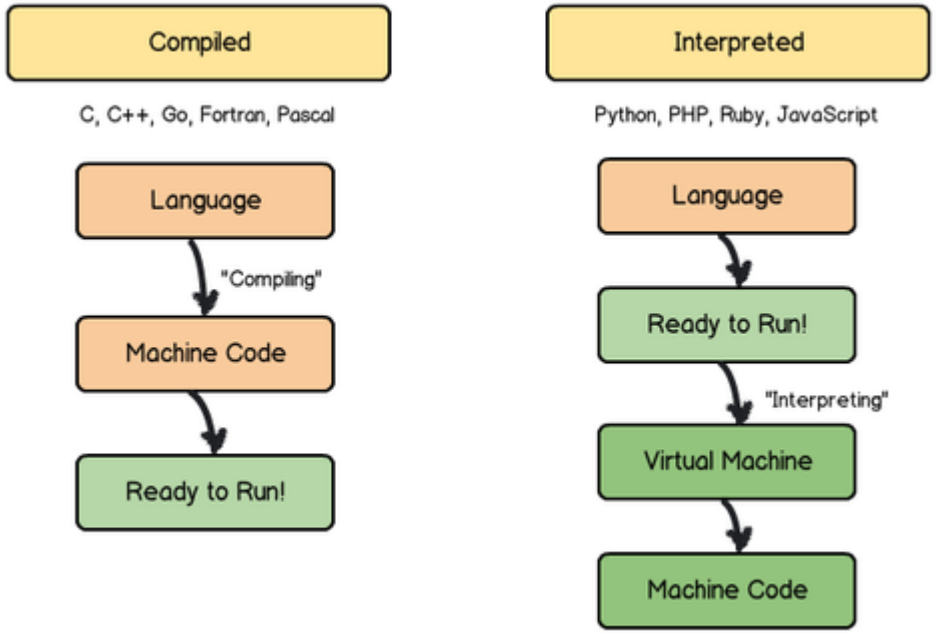
1) API와 Python 소개
2) 개발 환경 설정
   - 이베스트증권 Xing API 설치 및 DevCenter 접속
   - Anaconda 32비트 설치
   - PyCharm 에디터 설치
3) Python 기본 코딩 배우기 (Python vs. C# 비교 설명)
4) API 자동매매 코딩 순서
5) '서버접속'과 '로그인' 코드 작성 및 해설
6) '선물 현재가 시세받기' 코드 작성 및 해설
7) '선물 실시간 시세받기' 코드 작성 및 해설
8) '선물옵션 매도/매수 주문' 코드 작성 및 해설
9) PyQt 개요 및 설치, 사용방법

# 1) API 와 Python 소개

## 1) API 와 TR

API (Application programming interface) : HTS를 생각하면 된다. HTS가 일종의 API를 이용한 프로그램이다. 내가 만든 프로그램과 증권사 서버가 서로 데이터를 주고 받고 하면서 매매를 하는데, 이때 데이터를 서로 주고 받는 통로를 API 라고 한다

TR(Transaction) : 증권사로부터 필요한 데이터를 받을 수 있는 형식. TR을 사용하여 사전에 정해진 양식에 맞게 증권사 서버에 호출하면 서버에서 필요한 데이터를 받을 수 있다.




## 2) Python이란?

**프로그래밍 언어는 컴퓨터 시스템을 구동 시키는 소프트웨어를 작성하기 위한 형식 언어이다. (쉽게 말하면 컴퓨터에게 일을 시키기 위한 도구 중 하나가 프로그래밍 언어다)**

**Python 은 프로그래밍 언어 중에 script language 중 하나로써 문법이 쉽고 즉각적인 코딩 결과를 얻을 수 있으며, 외부 모듈이나 패키지 이용을 통한 응용력이 뛰어나 코딩 입문자들에게 추천된다. 최근들어 주식, 파생, 암호화폐 자동매매 분야에서도 널리 사용되고 있음**

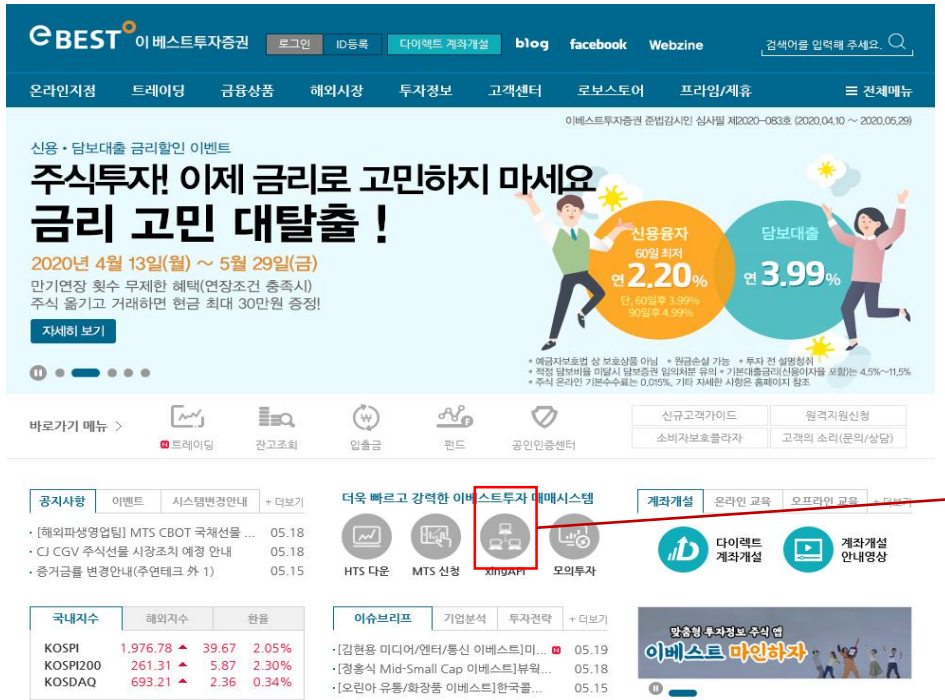
# 2-1) 개발환경설정 - 이베스트증권 XingAPI 등록신청 및 설치

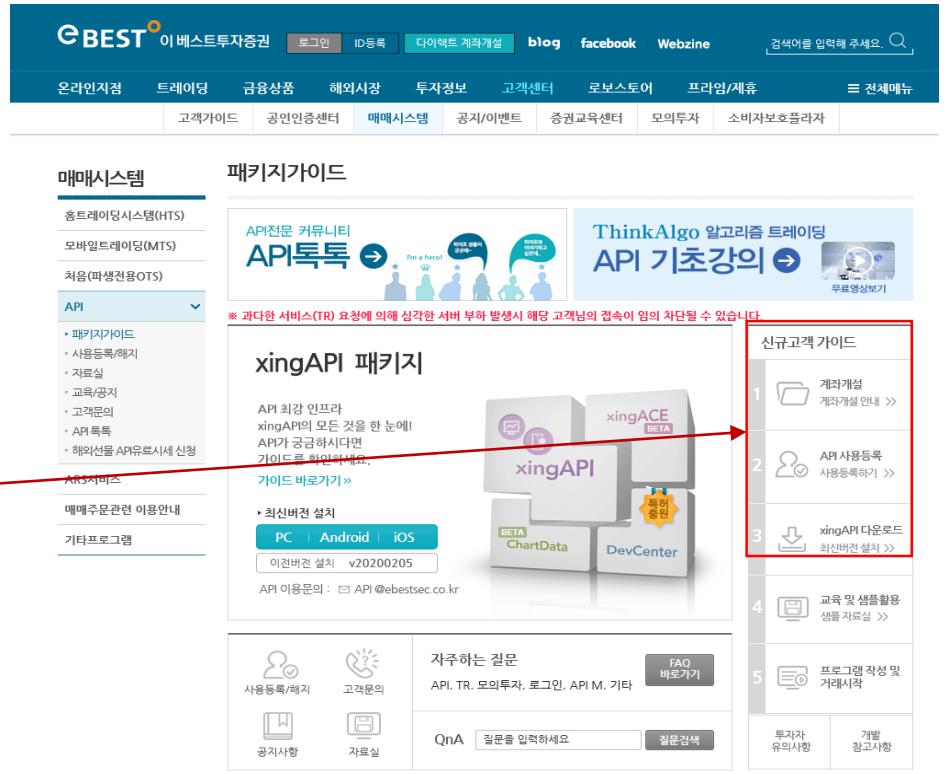
https://www.ebestsec.co.kr/

계좌개설 -> 이베스트증권 홈페이지에서 API 등록신청 및 다운로드 -> 모의투자 반드시 신청(프로그램 코딩 후 테스트는 모의서버에서 우선 진행)

XingAPI 설치와 설명 그리고 DevCenter 이용법 등 기초적인 API 관련 설명은 아래 '파생인의쉼터 API 자동매매 공지' 참고

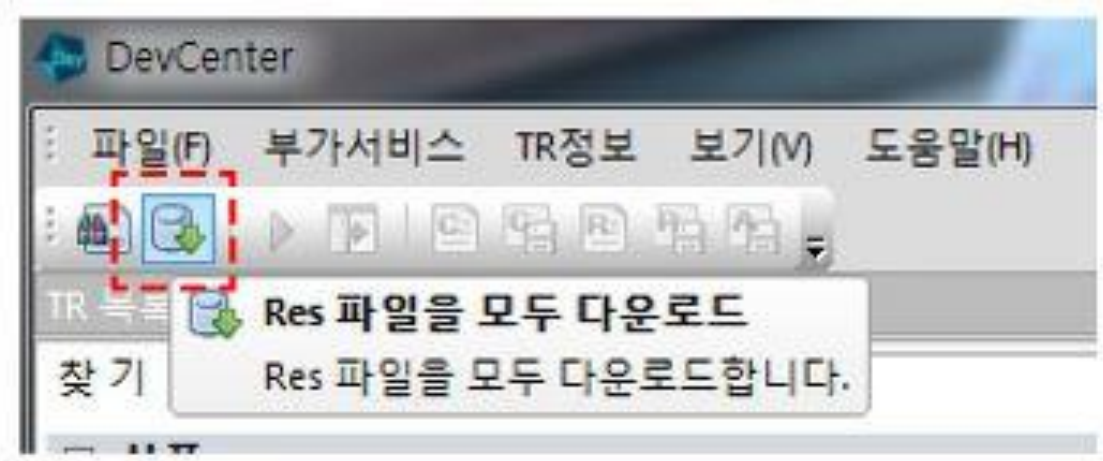
**https://cafe.naver.com/fo24/28895 (이베스트증권 XingAPI를 이용한 선물옵션, 파생, 주식 자동매매 가이드)**



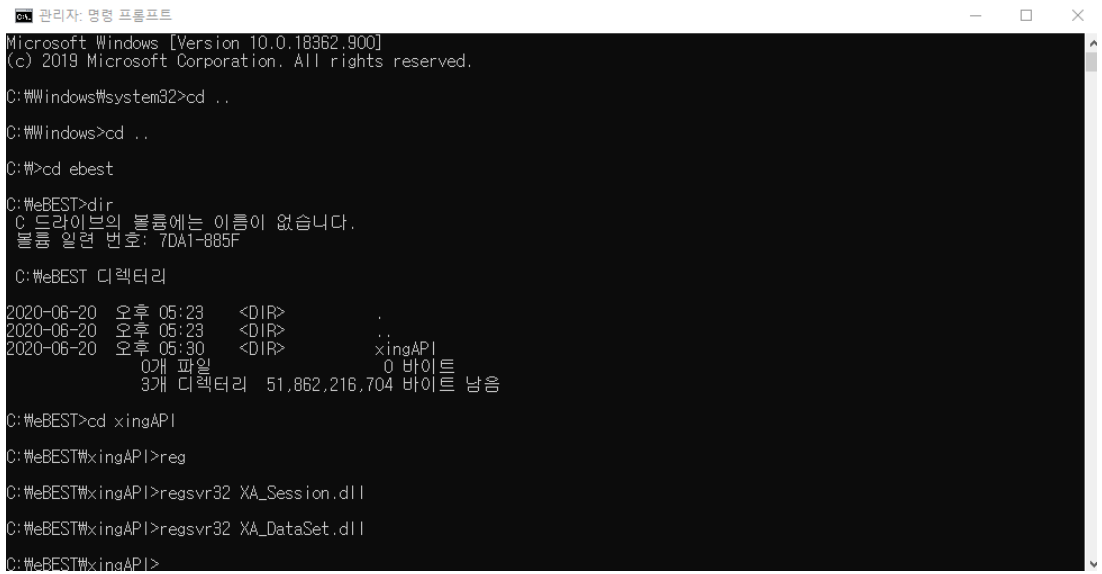



# 2-2) Res 파일 전부받기 및 Xing Session, Xing DataSet 등록

XingAPI 설치 완료하면 바탕화면에서 DevCenter 가 생성 -> 로그인해서 실행 -> **RES 파일 모두 다운로드**




VBA나 C#에서는 Xing 관련 COM 객체를 '참조'를 통해서 넣는 명확한 과정이 있지만, 파이썬에서는 그러한 과정이 없음. 필요한 경우 cmd 명령 프롬프트에서 관리자 권한으로 C:₩eBEST₩xingAPI₩Reg 등록




**원래 XingAPI를 설치하면 자동으로 Com 객체가 설치되어 따로 Reg 등록 할 필요는 없으나, 파일 실행시 com 오류 발생하면 등록필수!**

# 2-3) 개발환경설정 - 아나콘다(Anaconda)와 파이참(PyCharm)

## 1) 아나콘다란?

**Python만 설치하면 그 이후 필요로 하는 기능들을 사용하기 위해 다시 별도로 패키지들을 깔아야 한다. (API 자동매매만 하는 경우에도 파이썬 설치후 몇개의 패키지를 추가로 설치요함)**

**또한 각 패키지마다 Python과 버전이 맞는지도 확인해야 하는 등 이 과정이 매우 번거롭다.**

**'배포판'은 이걸 해결 하기 위해 Python과 기본적인 패키지가 같이 담겨져 있어서 이 걸 설치하면 위의 고민이 해결되는 것이다.**

**대표적인 배포판이 바로 'Anaconda' (약 400개의 패키지가 같이 깔림)**

**API 자동매매의 경우 아나콘다는 32비트로 써야 한다.(win32 Com 버전을 사용하기 위해서)**

**아나콘다 설명은 파생인의쉼터 관련 게시글 참고**

**https://cafe.naver.com/fo24/28053**

## 2) 파이참이란?

IDE (Integrated Development Environment) : 통합 개발 환경이라고 한다. 코딩, 디버그, 컴파일, 배포 등 프로그램 개발에 관련된 모든 작업을 도와주는 프로그램이다. 대표적인 IDE는 C# 프로그래밍을 위해서 사용하는 'Visual Studo 2019'가 있다.

C# 등과 달리 Python은 이런 코딩을 위한 공식적인 editor가 따로 없다. 메모장에다 코딩을 하며, 콘솔에서 바로 코딩을 하고 실행시킨다. 이런 점이 초보자들에게 파이썬을 어렵게 만든다. 그래서 '파생인의쉼터 카페'에서는 여러가지 종류의 Python 코딩 에디터 중에서 Pycharm을 초보자에게 기본 코딩 IDE로 추천하고 있다.

- 장점 : 무료이다, Python만을 위한 IDE이기 때문에 초보자가 쓰기 쉽다, 코딩과 디버깅을 하기 편하다, 파일과 함수 관리가 편하다. 다만 이런 장점에도 불구하고 사용을 위한 사전 설정을 해야 하고, 툴 자체에 대한 공부가 필요하다.

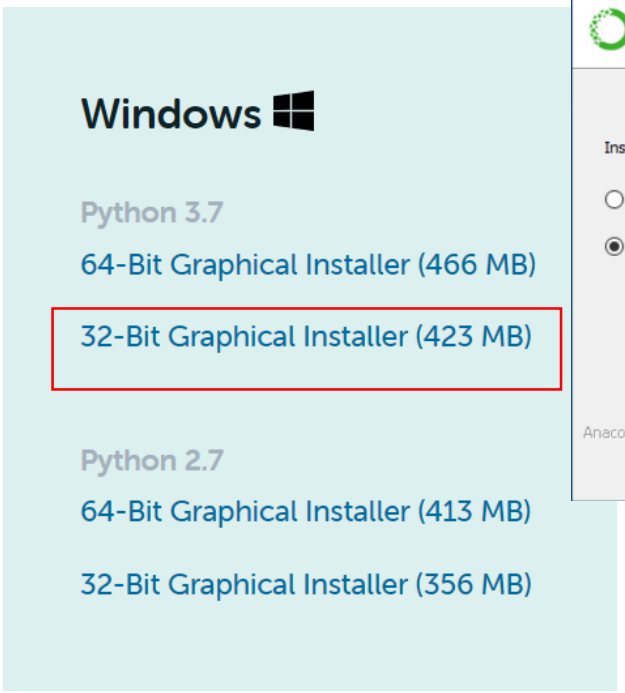
- 파이썬 코딩 에디터 종류 : Visual Studio Code, Atom, Spyder, Vi 편집기(리눅스와 비슷한 환경) 등등 수십가지가 있음

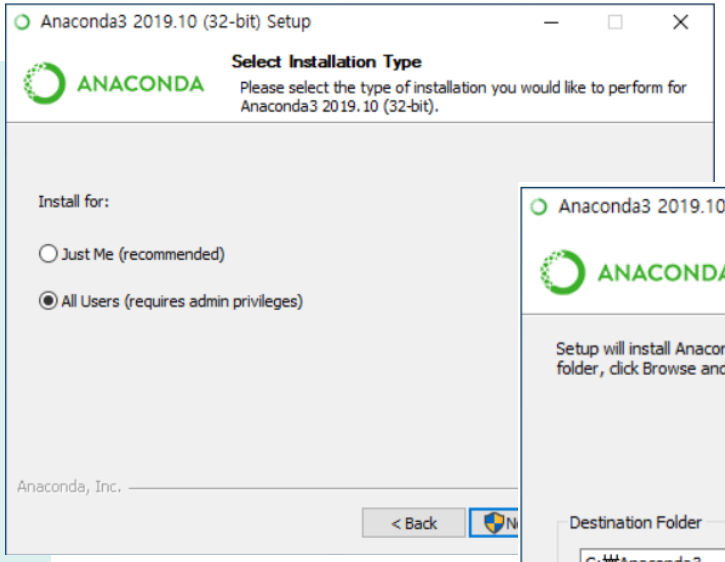
# 2-4) 개발환경설정 (Anaconda 32비트 설치)

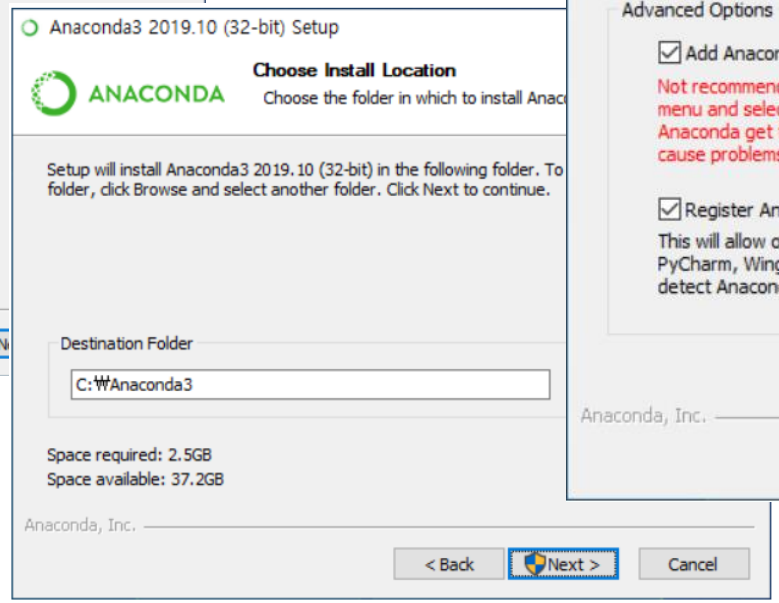
https://www.anaconda.com/products/individual

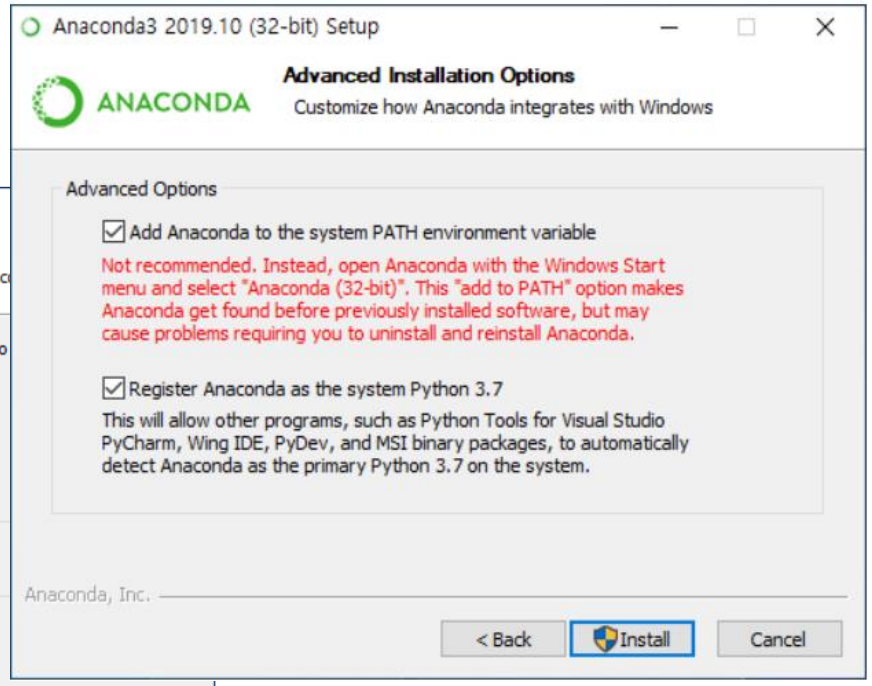
파이썬은 3.X 대로 사용 권장. 2.X 대는 유니코드(한글 깨짐) 문제 발생

**아나콘다32 비트 설치시 마지막 Advanced Options 에서 개발환경변수(Add Path) 등록 체크가 중요!!!**



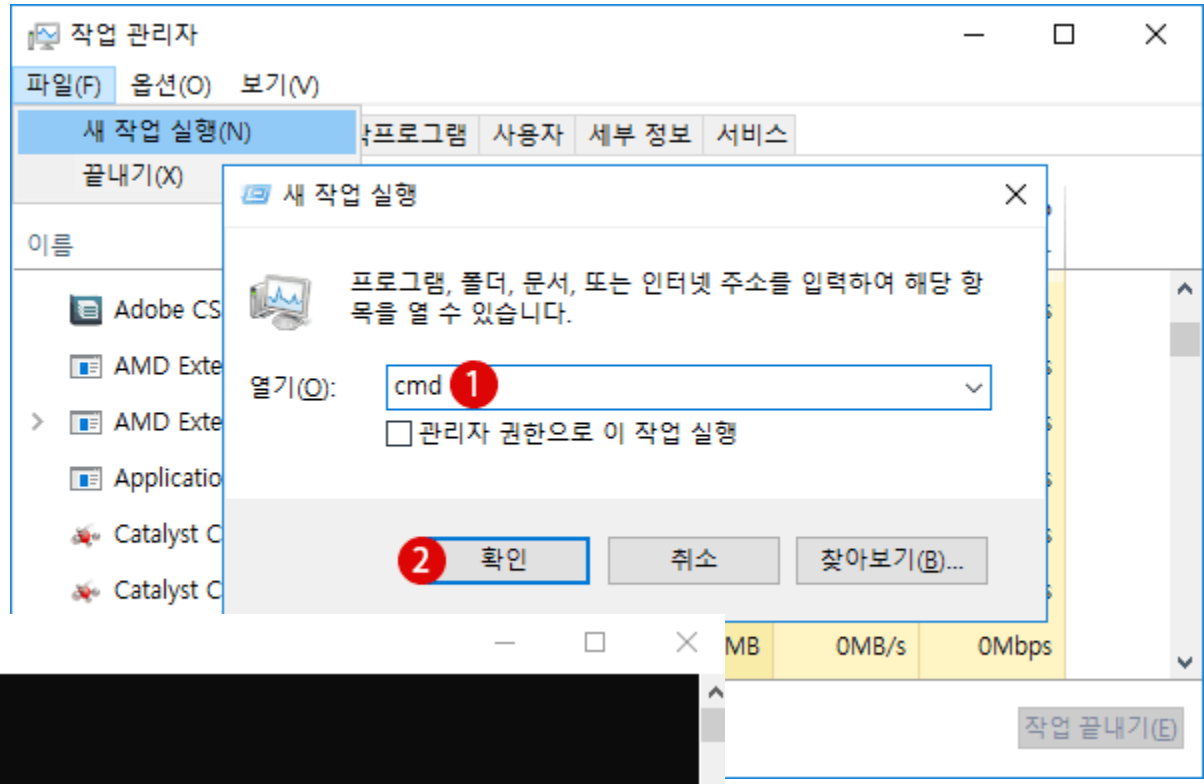



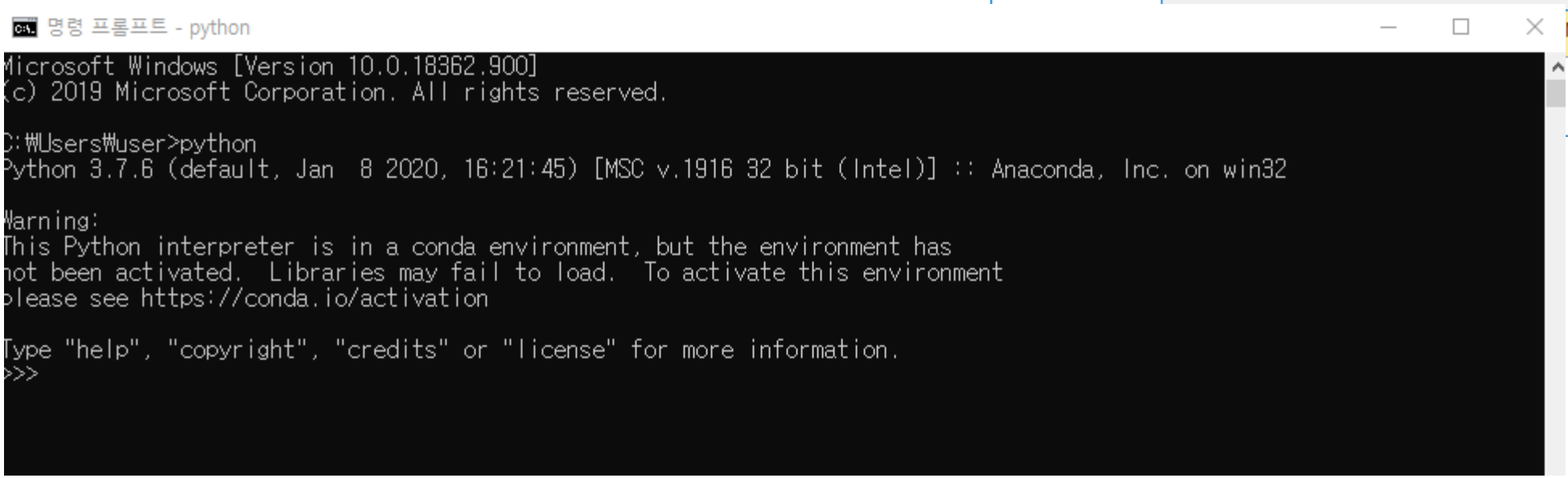



# 2-5) 아나콘다32 설치후 파이썬이 제대로 설치되었는지 확인

cmd(명령 프롬프트)에서 python 이라고 글자를 쓰고 엔터 치면 아래와 같은 화면이 나옴. 여기에 본인이 설치한 파이썬 정보가 나오면 제대로 설치됨

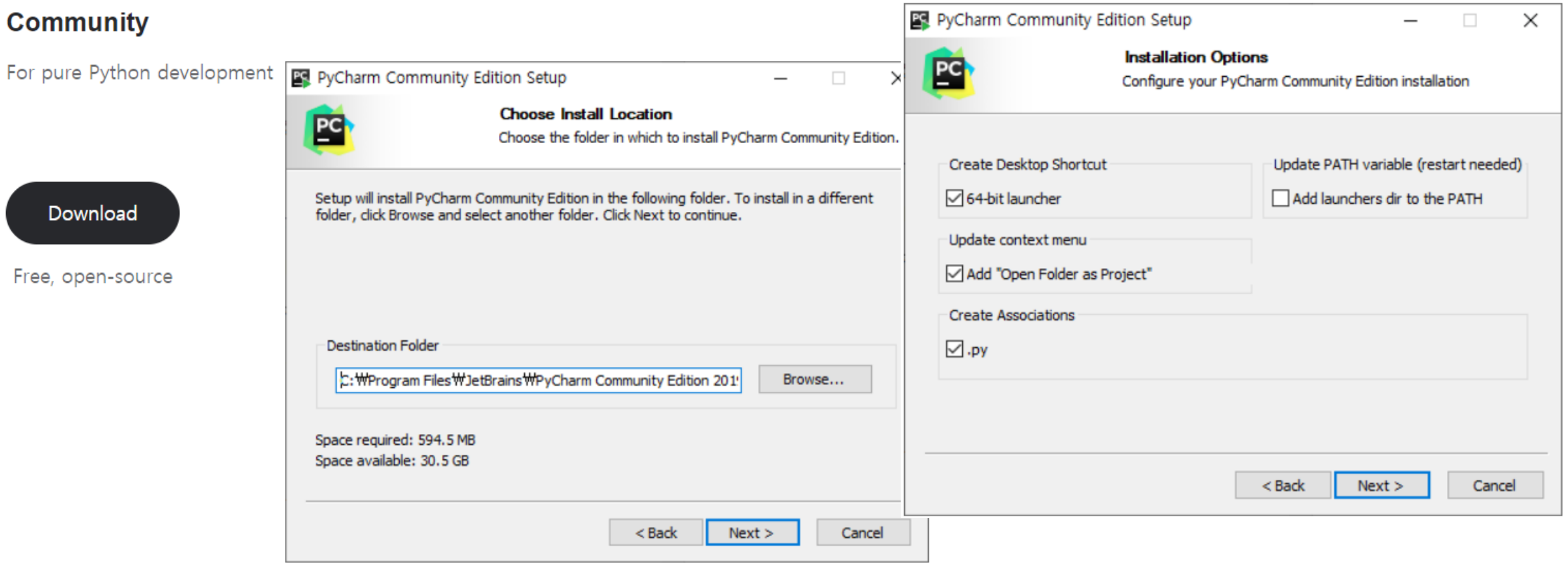
cmd 창을 **실행**하는 방법은 단축키 "윈도우 키+R" 이다. **실행** 창에 **cmd**라고 입력한뒤 확인 버튼을 누른다. 그럼 아래와 같이 명령 프롬프트(**cmd**)가 **실행**된다.



# 2-6) 개발 환경 설정 (PyCharm 에디터 설치)

https://www.jetbrains.com/pycharm/download/#section=windows

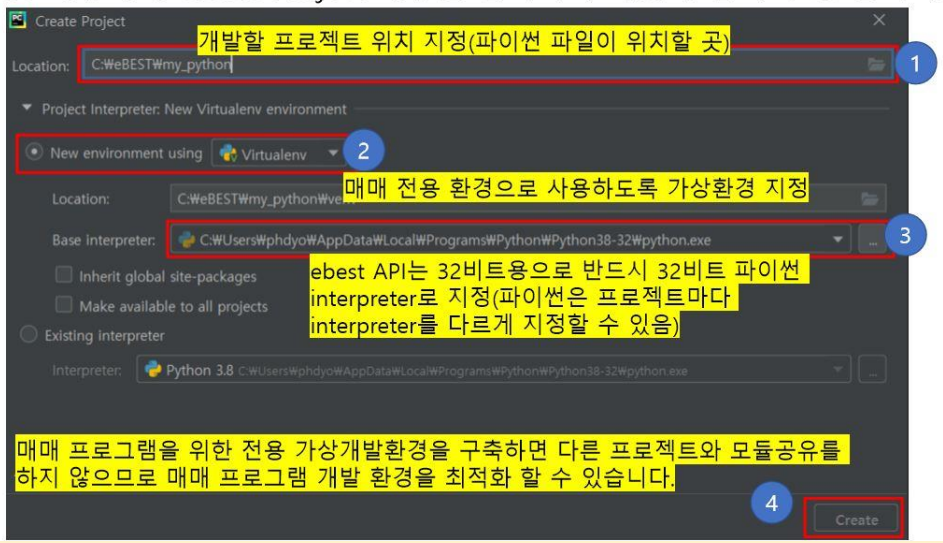


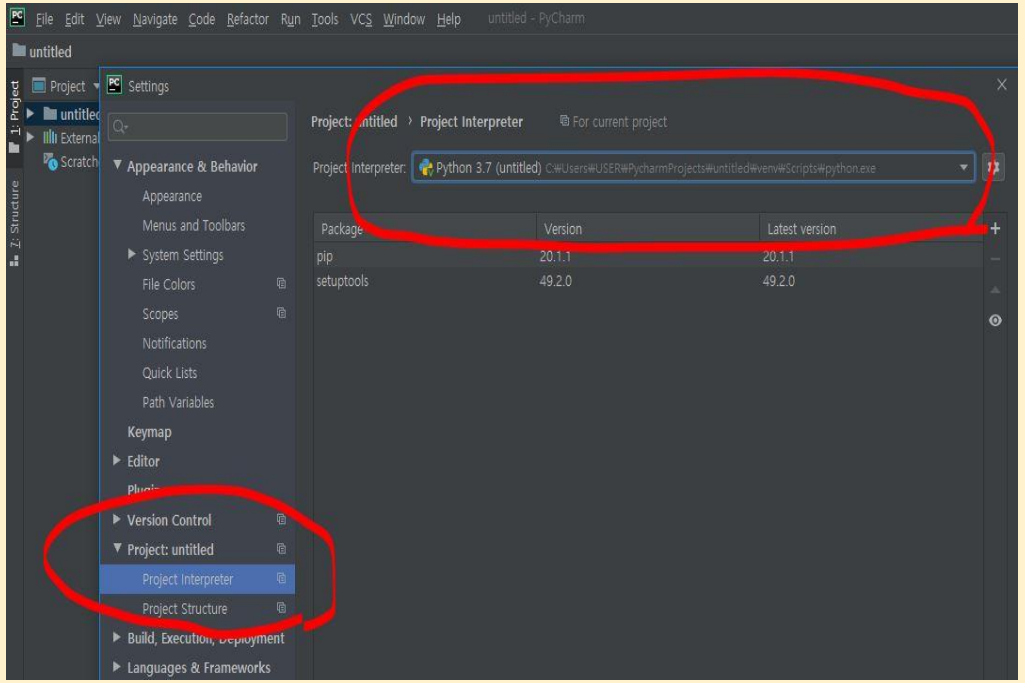
# 2-7) 개발 환경 설정 (PyCharm Editor setting)

**파이참 설치 후 꼭 해야 할 일이 바로 파이참 설정이다.**

## 파이참 개발환경 설정 (1) 전용 개발 환경 구성

- File 메뉴에서 Create Project 메뉴를 클릭하여 다음과 같이 구성 합니다.




**파이참 설정에 대한 보다 자세한 내용은 카페 관련 글 참고**

**https://cafe.naver.com/fo24/29204 <- eBest API 자동매매 개발을 위한 간편 파이썬 환경 설정(by 한강신)**

**https://cafe.naver.com/fo24/29274 <- 파이참에서 실행할 때 win32com.client 에러 발생시(by 말과의미)**

# 3-1) Python 기본 코딩 배우기

**1) Import 모듈 (C#에서는 using 에 해당)**

코딩 맨 상단에서 코딩. 파이썬 내장이 아닌 외부 프로그램 패키지나 모듈, 데이타등을 가지고올 때(impor는 수입하다는 뜻) 사용

Import win32com.client -> MS에서 만든 com 을 파이썬에서 활동하기 위해 가져온다는 뜻

```
import win32com.client #
import pythoncom # 파이썬에서 com를 사용하기 위해 사용
import time
```

**2) Class와 def**

a) class는 다른 언어와 마찬가지로 객체의 틀을 만드는 것임. 벽돌이라는 객체(object) 를 만들기 위해서는 먼저 벽돌틀인 class부터 만들어야 한다. 클래스는 일종의 선언 또는 정의라고 할 수 있다. 로그인이라는 class를 하나 만들면 계속 다른 곳에서 이 로그인이라는 class를 불러와서 사용할 수 있기에 개발자 입자에서는 코딩을 깔끔하게 할 수 있고, 오류가 생겨도 발견하기 쉽다. (C#에서도 클래스는 동일하게 사용)

class 클래스이름 : <- 클래스를 선언하는 코딩

b) def 는 definition 의 약자로, 함수를 정의하고 만드는 역할을 한다. 실제로 프로그래밍은 이런 함수들을 이용해서 실행을 하는 과정이다. class는 이런 함수들의 모임이고, class 안에 여러개의 def 들이 있는 셈이다. (C#에서는 함수 fuction을  메써드 method 라고 한다. 보통 매서드는 클래스 내에 정의된 함수를 말한다 )

def 함수이름(): <- 함수를 선언하는 코딩

```
class 로그인:
    로그인여부 = False

    def OnLogin(self, code, msg):
        if code == "0000":
            print("로그인 성공")
            로그인.로그인여부 = True
        else:
            print("로그인 실패")

    def WaitResponse(self):
        while not 로그인.로그인여부:
            pythoncom.PumpWaitingMessages()

    def LoginRequest(self):
        self.Login(아이디, 비밀번호, 공인인증번호, 0, 0)
        self.WaitResponse()
```

**3) If ~ eles 구문**

실제 자동매매 전략을 돌릴 때는 어떤 조건에 맞으면 작동해라.

이런 형태로 코딩을 하게 된다.

즉 if 고가 > 저가 then 매수하라 else 대기하라

파이썬에서는 아래처럼 코딩된다

If high>low:

Buy() <- Buy는 함수형태로 미리 만든 것을 불러와서 사용

Else

Hold() <-hold 역시 함수형태로 미리 만든것 불러와서 사용

```
## 전략구동 시작
while 주문회수 <= 최대주문횟수:
    t2101.QueryRequest()
    현재가 = t2101.GetCurrentPrice()
    시가 = t2101.GetOpenPrice()
    if 현재가 > 시가:
        주문클래스.OrderRequest(1,현재가,1)
        주문갯수 -= 1  ## 주문갯수 = 주문갯수 - 1
    else:
        주문클래스.OrderRequest(2,현재가,1)
        주문갯수 += 1 ## 주문갯수 = 주문갯수 + 1
    주문회수 += 1
    time.sleep(60*작동분단위)
```

# 3-2) Python으로 코딩할 때 주의사항

**1) 파이썬은 코딩순서로 실행된다. 실행되면서 객체가 생성된다. 그래서 코딩순서가 중요하다.  일반적으로 선언(Class)이 가장 앞에  둔다. 그리고 객체 생성(Dispath)이 두번째로 가야 한다.**

**2) 기호 : C#에서는 같음은 == 또는(or)는 II 이런 연산자로 사용함. 파이썬에서는 같음은 &, or는 I 로 표시. 파이썬에서 ==는 같음의 뜻(파이썬에서는 더하기 +, 곱하기 * 배기 – 나누기 / )**

**3) 스펠링과 대문자와 소문자 : 스펠링을 틀리거나 대문자와 소문자 구별하지 않은 경우 오류 발생(어느 코딩에서는 대소문자 구별은 중요함)**

**4) 주석 앞에는 #을 붙인다. <-> C#에서는 // 으로 주석처리**

**5) print을 할 때는 변수() 묶는다. 문자로 나타내려면 (“내용”)으로 해야 한다. print(“파생인의쉼터 화이팅!”). Print(10+30)은 덧셈이 되어서 40이 결과값으로 나온다.**

```
>>> print(10+30)
40
>>> print("10"+"30")
1030
>>> print("파생인의쉼터 화이팅!")
파생인의쉼터 화이팅!
```

*** 파이썬에서의 print()는 C#에서는 Console.WriteLin();**

**6) 띄어쓰기 매우 유의 : 많은 사람들이 Python을 쓸 때 하는 실수이다.**

**Python은 def, class, if/else, while 등을 구분할때 띄어쓰기를 사용한다Tab을 사용하거나 스페이스바를 4번 사용해야한다**

# 3-3) Python vs C# 비교

- **Python (대표적인 script langauge, complie 과정 없이, 코딩을 runt 하면 한줄씩 interpreting 방식으로 기계어로 변환되어 실행)**
  - 스크립트 랭귀지 특성상 문법이 쉬운 편. 매매전략 아이디어가 떠올랐을 때 빠르게 코딩해서 확인하는데 제격(한줄 코드 작성후 바로 실행해서 결과 확인 가능) . **GUI에서 form 등을 중요하게 생각하지 않고 console에서 처리하는데 제격인 언어. PyQt를 이용해서 form을 만들수는 있음.**
  - 그러나 스크립트 랭귀지는 속도가 느린 언어이고, 코드상의 에러는 실행시에 발견됨. 실행하면서 코드 수정 요함. 또한 공식적인 에디터 등이 따로 없음. 개발환경설정하는 다소 어려움이 있음.
  - Python만으로는 할 수 있는게 없음. 외부 모듈이나 패키지를 적극 활용해야 함. 다만 최근 금융, 주식, 투자 관련 참고할 코딩 소스(Source)나 라이브러리(lib), 예제가 늘어나고 있고, 머신러닝 등을 하는데 있어 필수 언어 다만 파이썬 버전과 외부 모듈이나 패키지 버전을 잘 맞추어야 에러가 안생김
- **C# (대표적인 complied lauguage, 코딩 후 컴파일 과정을 거쳐 기계어로 변환 후 실행됨)**
  - Visual Studio 라는 강력한 IDE(코딩, 디버깅, 컴파일 등) 를 사용. 기본 라이브러리 지원이 잘 되어 있어 기능들을 쓰기가 편하다**. 특히 Form 등을 만들어서 사용할 때는 C#을 추천함.**
  - 일반인 대상으로 접근성이 안좋음(개발자용 개발툴을 활용 ,Visual Studio 숙련에 시간이 필요). 언어 활용 복잡함(다양한 변수 타입, 규모가 큰 개발 환경에 필요한 모듈 활용) <-> 파이썬은 상대적으로 코딩이 쉬움( 변수 타입 구분 없고, C#에 비해서 적은 타이핑 환경) 그래서 상대적으로 코딩 입문자들은 Python 을 선호함 (보통 대학 프로그래밍관련 전공파트에서는 C#은 정규과정에 있으나, Python은 따로 가르치지 않음)
  - 단점은 매매전략과 관련해서 일부 전략은 직접 코딩해야 하는데 이게 만만치 않음 <->Python은 기존에 만들어진 라이브러리를 불러와서 사용하면 그만

**Form을 중심한다면 C#, 빠른 개발 테스트가 필요하다면 pyhon. 분봉 단위 매매라면 Python, 1초 단위 매매라면 C#, 각각의 장단점이 있기에 둘 다 배워두면 좋음!**

# 3-4) Python vs. C# 코드 비교

## 로그인 코드 함수 정의 (void~ / def ~) 부분 비교

C#

```
void 로그인_접속(string szCode, string szMsg)
{
    Console.WriteLine("szCode " + szCode + ", szMsg " + szMsg);

    if (szCode == "0000")
    {
        Console.WriteLine("로그인  성공");
    }
    else
    {
        Console.WriteLine("로그인  실패");
    }
}
```

Python

```
def OnLogin(self, szCode, szMsg):
    print("szCode " + szCode + ", szMsg " + szMsg)
    if szCode == "0000":
        print("로그인 성공")
    else:
        print("로그인 실패")
```

| | C# | Python |
|---|---|---|
| 작업의 구분 | 중괄호 { } 사용 | Tab을 이용한 들여쓰기 사용 |
| 변수 선언 | string과 같이 데이터를 구분 | 단순히 이름만 선언 |
| Self | 파이썬에서는 함수를 만들때 def 에서 첫번째 인자로 self(변수의 일종) 사용이 원칙임<br>**Self의 의미는 해당 class내에서 onLogin을 불러온다. Self 생략하면 다른 곳의 onLogin 호출됨** | |
| 출력 | Console.WriteLine() | print() |

# 4) API 자동매매 코딩 순서 (매매할 때를 떠올려보자)

1) **9시 이전에 HTS 접속 => API에서는 서버접속과 로그인 코드 작성**

2) **최근월 코스피200 선물 코드 조회 -> API에서는 지수선물마스터 t8432 TR 이용해서 조회**

3) **선물 현재가 조회 -> API에서는 선물옵션 현재가(시세)조회 t2101 TR 이용해서 조회**

4) **선물 실시간시세 조회 -> API에서는  KOSPI200 선물체결 FC0 REAL(실시간TR) 이용해서 조회**

5) **주문하기 -> API 에서는 주문TR 이용해서 매수/매도 주문 (선물옵션정상주문  TR CFOAT00100 이용)**

6) **잔고조회 -> 주문내역(체결/미체결 등)은 HTS에서 잔고조회 (초보자과정에서는 생략)**

**TR = transaction 약자**

* TR는 증권사 서버와 내가 만든 프로그램간에 데이타를 송수신하는 행위 자체
* DevCenter에 TR 목록이 있듯이, 일회성 데이타를 주고 받을 때 사용 (**현재가 TR, 주문 TR 등**)

**Real = 실시간TR 라고도 많이 부름**

- TR과 다르게 데이타를 한번 요청하면 계속해서 데이타가 송수신. **대표적으로 실시간 시세 받기가 Real**

# 4-1) DevCenter 활용법과 TR 레이아웃


**TR은 사전에 이미 정의된 형식(값)을 입력해서 서버에 전송하고, 역시 사전에 정의된 결과값을 수신한다. 이렇게 사전에 정의된 형식 (값)을 TR Layout(TR 배열) 이라고 한다.**

**TR Layout은 1) 이베스트증권에 데이타를 요청할때 사용하는 형식(값)인 InBlock(입력값)과 2) 이베스트증권에서 데이타를 수신할 때 사용하는 OutBlock(출력값) 있다.**

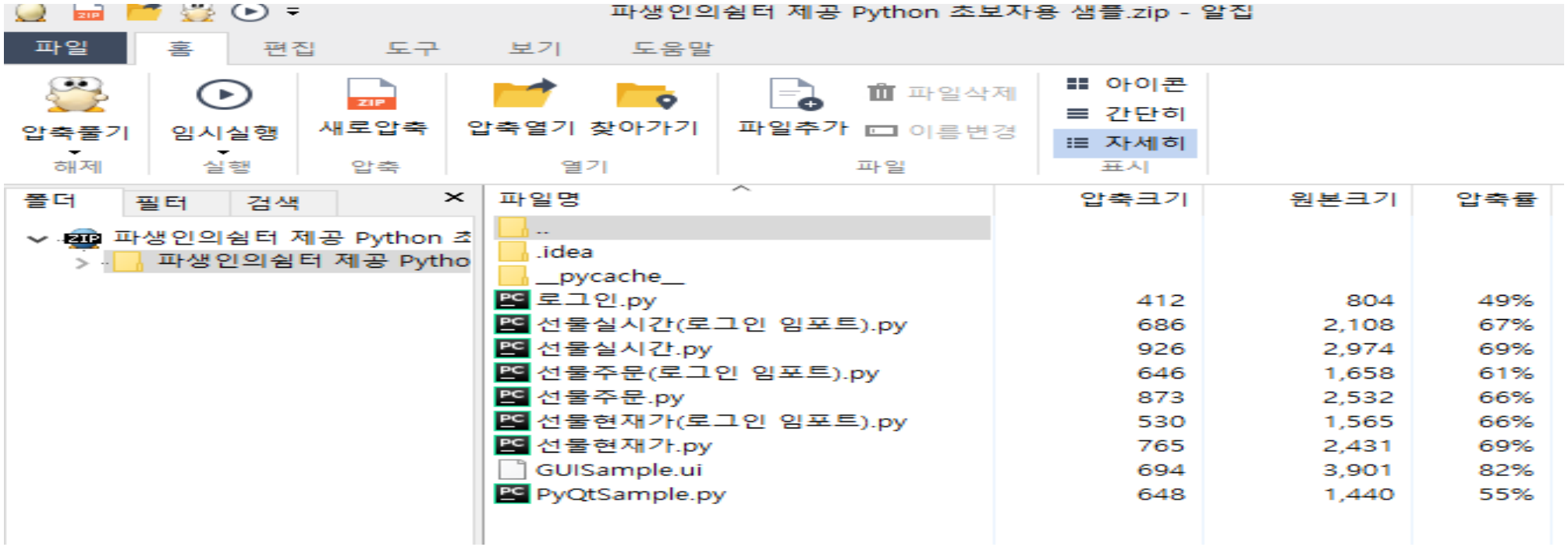
**이베스트증권 서버에서 어떤 데이타(TR, REAL)를 주는지, 그리고 TR Layout(InBlock, OutBlock)은 어떻게 되는 Devcenter에서 확인**

**Res는 TR layout을 com 버전에서 인식할수 있는 형식으로 변경한 구조. Com 버전에서는 res를 꼭 입력해야만 작동되며, RES 파일은 TR Layout을 파일로 저장한 것**

**** 앞에서 Devcenter 에서 모든 res 파일 다운로드 받으라고 한 것도 이것 때문임. 사용하고자하는 res 파일이 내 컴퓨터 적절한 폴더에 없으면 데이타 송수신 불가.**

**** TR 레이아웃은 복사해서 메모장 등에 붙여넣고 사용해야 오타가 없음(대문자 소문자 섞여 있음) Camel Case란 단어가 합쳐진 부분마다 맨 처음 글자를 대문자로 표기하는 방법. 마치 낙타 등처럼 글자가 중간에 튀어나옴. 이유는 컴퓨터는 띄어쓰기 인식 못함. OrdNo (주문번호)**

# 4-2) Python 샘플 파일 설명




**파생인의쉼터에 첨부된 압축파일을 풀면 위와 같은 샘플 파일이 있음.**

**각 샘플파일에서 아이디, 접속비번, 공인인증 등은 본인의 것으로 수정해서 사용해야 함.**

**선물현재가.py는 로그인 정보+ 선물 현재가 요청이 통합된 형태**

**선물현재가(로그인 임포트).py는 로그인 정보를 import 해서 불러오는 분리형**

# 이베스트증권 XingAPI

# 초보자용 Python 개발가이드

# 제1장 서버접속과 로그인 코딩

API란 증권사 서버와 내 프로그램간에 데이타를 주고받는 것을 말하며,
증권사 서버와 데이타를 주고 받기 위해서는 가장 먼저 증권사 서버에 접속을 하고,
내 아이디/접속비밀번호/공인인증을 전달해서 로그인을 해야 한다.

그리고 **로그인을 해야만 비로소 시세를 받거나 주문을 할 수 있기에 서버접속과 로그인이 가장 먼저 해야할 일이다.**

# 5) 서버접속 및 로그인 코드(로그인.py)

```
import win32com.client
import pythoncom


class 로그인:
    로그인여부 = False  # false 거짓이면 현재 로그인이 되어 있지 않은 상태, true 면 이미 로그인 된 상태

    def OnLogin(self, szCode, szMsg): # OnLogin 함수가 호출되었을 때 에러코드와 메세지를 받아옴
        print( " szCode  "  + szCode +  " , szMsg  "  + szMsg) # 에러코드와 메세지를 출력합니다
        if szCode == "0000": # 만약 에러코드가 "0000" 이면
            print("로그인 성공") # 로그인 성공을 콘솔 화면에 출력
            로그인여부 = True
        else: # 그외라면
            print( " 로그인 실패 " ) # 로그인 실패를 콘솔 화면에 출력

접속 = win32com.client.DispatchWithEvents("XA_Session.XASession", 로그인) # 접속과 관련된 일을 하는 객체를 만듭니다

아이디 = "" # 내 아이디
비밀번호 = "" # 내 비밀번호
공인인증번호 = "" # 내 공인인증번호

접속.ConnectServer("demo.ebestsec.co.kr", 20001) # 서버에 접속을 시도 (실거래서버는 hts.ebestsec.co.kr)
접속.Login(아이디, 비밀번호, 공인인증번호, 0, 0) # 로그인을 시

# 로그인이 성공하지 않았으면 계속 응답을 기다립니다
while 로그인.로그인여부 == False: # 로그인여부가 False 즉 로그인이 성공하지 않았으면 while 이하를 반복합니다
    pythoncom.PumpWaitingMessages() # 응답이 올때까지 기다려라
```

**#은 주석을 달 때 사용합니다. 주석은 코드를 설명하기 위한 수단으로서 코드 실행에 아무 영향도 주지 않습니다.**

# 5-1) 서버접속과 로그인 코드 해설

```
import win32com.client
import pythoncom

class 로그인:
    로그인여부 = False

    def OnLogin(self, szCode, szMsg):
        print("szCode " + szCode + ", szMsg " + szMsg)
        if szCode == "0000":
            print("로그인 성공")
            로그인여부 = True
        else:
            print("로그인 실패")
```

Import 설명 : win32com 과 pythoncom(파이썬과 com을 연결하는 역할) 모듈을 현재 Python 코드에 가져온다. 코딩 가장 앞 부분에서 써줘야 함

**파이썬에서는 항상 먼저 Class를 만들어서 사용하며 Class 부분이 앞에 와야함. 그러다보니 요청하는 코딩은 순서상 하단에 있고, 먼저 앞에 수신하는 부분에 대한 코딩이 class로 먼저 옴.** 로그인 **이 Class 이름. Class로 만들면** 로그인 을 계속 다른 곳에서 불러서 사용 가능

OnLogin은 이베스트 증권에서 정해놓은 메써드(VBA나 C#에서는 Login)로 로그인 이후 서버에서 응답이 왔을 때 실행

**Login 대신 OnLogin을 사용하는 이유 :** 파이썬이나 C 언어에서 com버전을 사용할 경우 데이터나 메시지나 받을 때(Receive) 관련 부분에서 코딩시 on을 붙여야만 작동된다. On 안붙이면 동작을 안함

szCode는 이베스트에서 로그인에 대한 결과로 준 코드이고 szMsg는 메시지

이베스트 증권이 준 결과값(코드와 메시지)을 출력하고 "0000" 일 때 로그인 성공을 출력하고 로그인여부를 True로 바꿈 (코드 "0000"은 로그인이 정상적을 되었을 때 결과) 아니라면 로그인 실패를 출력

# 5-2) 서버접속과 로그인 코드 해설

```
접속 = win32com.client.DispatchWithEvents("XA_Session.XASession", 로그인)
```

**# DispatchWithEvents 설명 : XA session에 이벤트가 발생하면 로그인 class로 해당정보를 전달해라. 접속에서는 접속과 관련한 처리를 해주는 객체를 만들고, xa session.dll. Dataset.dll을 등록해주는 역할**

COM을 사용하기 위해서 win32com.client를 import하고 win32com.client.DispatchWithEvents()로 COM 객체를 만든다. 그러면 DispatchWithEvents()에 2개의 인자가 전달된다. 첫번째는 COM이름, 두번째는 이벤트 콜백시 호출될 클래스명이다. 여기서 클래스 이름은 로그인 이다.

**XingAPI는 3개의 COM이 제공 : XA_Session.XASession, XA_DataSet.XAQuery, XA_DataSet.XAReal**

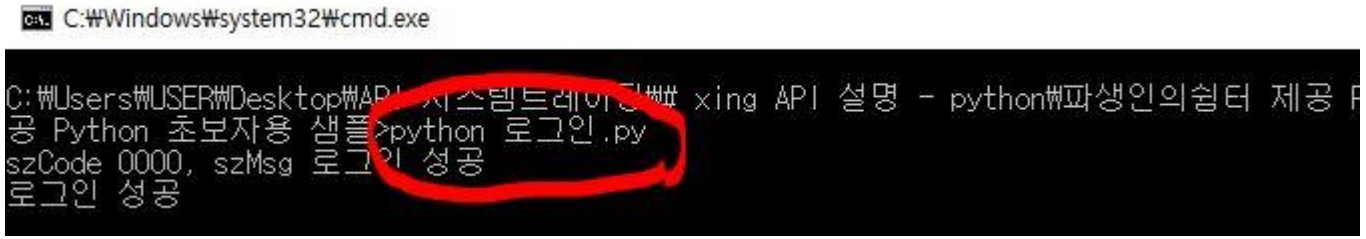
```
아이디 = "" # 아이디 넣어라
비밀번호 = "" # 비밀번호 넣어라. 모의에서는 0000
공인인증번호 = "" # 공인인증번호 넣어라. 모의에서는 불필요
접속.ConnectServer("demo.ebestsec.co.kr", 20001) # 모의서버에 접속
접속.Login(아이디, 비밀번호, 공인인증번호, 0, 0) #로그인을 시도

while 로그인.로그인여부 == False: # 로그인여부가 False 즉 로그인이 성공하지 않았으면 아래를 반복. While은 반복문
    pythoncom.PumpWaitingMessages() # 응답을 기다림. 즉 로그인을 성공하지 않으면 계속 로그인될때까지 기다림
```

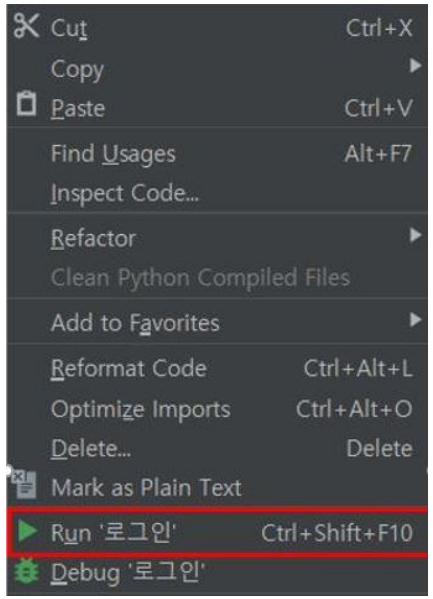
**** 파이썬 언어의 고유의 특성. 이벤트를 기다리는 코드 .**로그인여부가 거짓이라면(로그인이 안되어 있으면) 프로그램이 종료되지 않고 응답이 오기를 기다린다는 코드. 응답이 오기를 기다려야 하는 이유는 XingAPI가 비동기 방식이기 때문. 만약 이 코드를 쓰지 않으면 로그인 요청한 후에 프로그램이 끝나버려 응답을 받을 수 없음

# 5-3) 로그인.py 실행방법과 실행결과

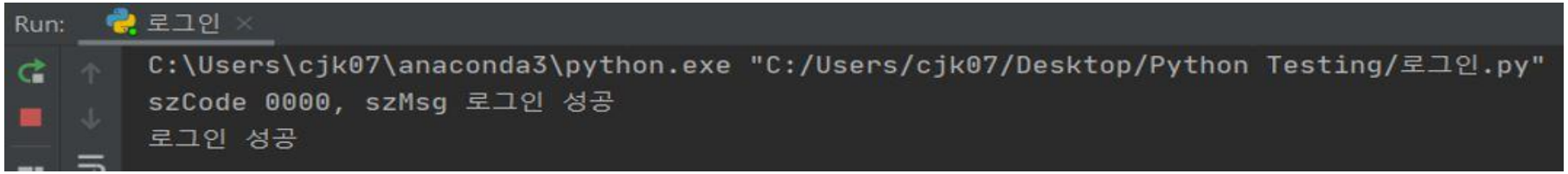
*** 파이썬 코드로 작성된 코딩의 실행방법에는 크게 2가지 방식이 있다.**

**1) 가장 일반적인 방식으로 '실행할 파이썬 파일 ' 이 저장되어 있는 폴더에서 shift + 마우스 우축 클릭을 한 후 '여기서 명령창 열기 ' 로 cmd프롬프트 창을 연 뒤 python 파일이름.py 를 입력하는 방법이다 (파이썬은 해당 파일이 있는 디렉토리에서 실행 또는 해당 디렉토리 주소를 정확히 기재해야 작동됨)**




**2) 파이참에서 로그인.py에서 프로젝트로 부르거나 또는 파이참을 켜고 로그인.py를 마우스로 당겨서 가져다 놓고, 화면 왼쪽 상단에 '로그인' 파일 이름을 마우스로 우클릭하면 왼쪽처럼 run '로그인' 할 수 있는 메뉴가 나온다. 여기서 실행 하면 화면 하단처럼 로그인.py 파일이 실행된다.**



# 이베스트증권 XingAPI

# 초보자용 Python 개발가이드

# 제2장 선물 현재가 받기 코딩

XAQuery 이용, TR목록 t2101 , 해당 TR 목록은 DevCenter에서 확인

가장 많이 사용되는 TR이며, 선물 현재가를 확인할 때 사용
(한번 요청하면 그 순간 선물 체결가를 한번만 결과값으로 가져옴. 따라서 계속해서 현재가 시세를 받고 싶으면 실시간 현재가 REAL 을 받아야 함)

# 6-1) 선물 현재가 코드(로그인+선물 현재가 수신 통합형, 선물현재가.py)

```
import win32com.client
import pythoncom

class 로그인:
    로그인여부 = False

    def OnLogin(self, code, msg):
        if code == "0000":
            print("로그인 성공")
            로그인.로그인여부 = True
        else:
            print("로그인 실패")

class 선물현재가:
    수신여부 = False # 선물현재가를 현재 수신하지 않을 경우에 false

    def OnReceiveData(self, code):
        선물현재가.수신여부= True  # 현재가 수신을 했다면 아래처럼 화면에 출
력

        print("종목명 : " + t2101.GetFieldData("t2101OutBlock", "hname", 0))
        print("현재가 : " + t2101.GetFieldData("t2101OutBlock", "price", 0))
        print("전일대비구분 : " + t2101.GetFieldData("t2101OutBlock", "sign",
0))
        print("전일대비 : " + t2101.GetFieldData("t2101OutBlock", "change",
0))
        print("등락률 : " + t2101.GetFieldData("t2101OutBlock", "diff", 0))
        print("거래량 : " + t2101.GetFieldData("t2101OutBlock", "volume", 0))
        print("상한가 : " + t2101.GetFieldData("t2101OutBlock", "uplmtprice",
0))
        print("하한가 : " + t2101.GetFieldData("t2101OutBlock", "dnlmtprice",
0))
        print("시가 : " + t2101.GetFieldData("t2101OutBlock", "open", 0))
        print("고가 : " + t2101.GetFieldData("t2101OutBlock", "high", 0))
        print("저가 : " + t2101.GetFieldData("t2101OutBlock", "low", 0))
```

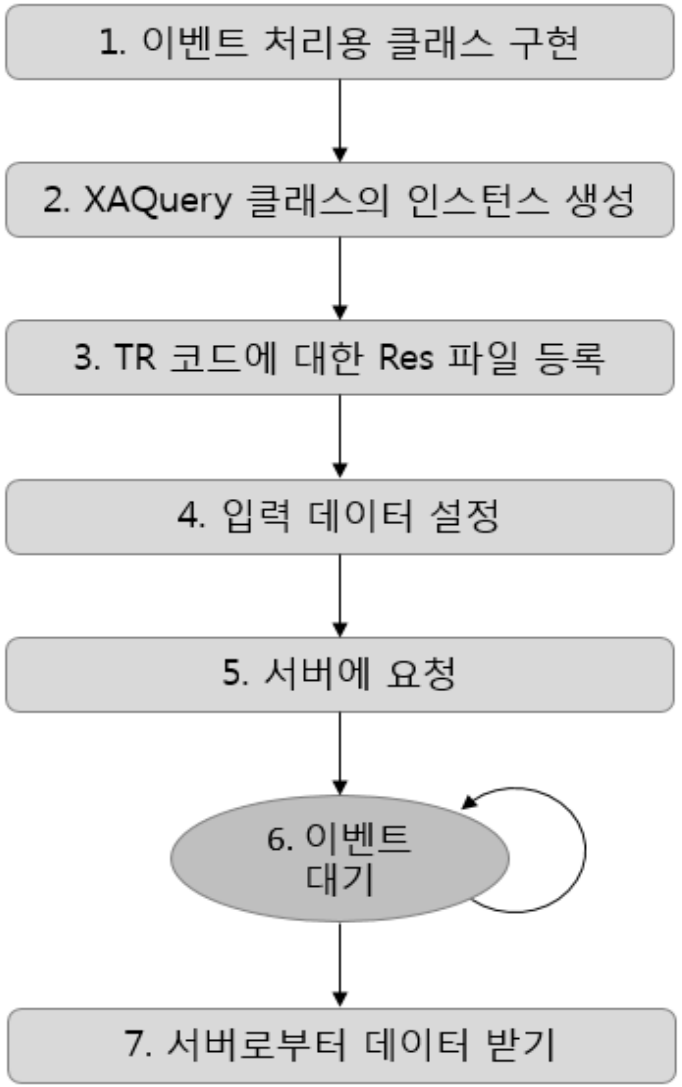
```
# ------------------------------------------------------------------------
# login
# ------------------------------------------------------------------------
세션 = win32com.client.DispatchWithEvents("XA_Session.XASession", 로그인)

아이디 = "" # 내 아이디
비밀번호 = "" # 내 비밀번호
공인인증번호 = "" # 내 공인인증번호

세션.ConnectServer("demo.ebestsec.co.kr", 20001)
세션.Login(아이디, 비밀번호, 공인인증번호, 0, 0)

while 로그인.로그인여부 == False:
    pythoncom.PumpWaitingMessages()

# ------------------------------------------------------------------------
# t2101
# ------------------------------------------------------------------------
t2101 = win32com.client.DispatchWithEvents("XA_DataSet.XAQuery", 선물
현재가)

t2101.ResFileName = "C:₩₩eBEST₩₩xingAPI₩₩Res₩₩t2101.res"

t2101.SetFieldData("t2101InBlock", "focode", 0, "") # 종목코드

t2101.Request(False)

while 선물현재가.수신여부 == False: #선물현재가를 현재 수신 받지 않고 있다
면
    pythoncom.PumpWaitingMessages()
```

로그인 코드가 같이 있는 이유 - 모든 TR을 사용하기 위해서는 서버에 로그인이 되어 있어야 함. HTS를 통해 로그인을 하지 않으면 서비스를 이용하지 못하는 것 처럼 API를 통해 TR을 불러오는 것 역시 로그인을 먼저 해야 이용할 수 있음. 로그인은 항상 모든 API 코딩 앞부분에 존재

TR에서는 수신여부 false 일 때 선물현재가.수신여부= True 가 더 들어감(REAL 에서는 안들어감)

* 위 코딩에는 Receive Message 부분이 생략됨 : Tr 조회가 안되는 이유에 대한 메시지를 받는 부분(시간제한 등 기타 사유로 블락될 경우)

# 6-2) 파이썬에서 TR(단일 데이터 조회) 코딩 순서




**def OnReceiveData**(self, code)
**# 데이터 수신 이벤트 먼저 설정. 요청부분은 아래에 있음. Class부터 먼저 코딩.** 이베스트투자증권의 서버는 TR 처리가 완료되면 OnReceiveData 메서드를 콜백

```
t2101 = win32com.client.DispatchWithEvents("XA_DataSet.XAQuery", 선물현재가)
```

**# DispatchWithEvents 함수를 사용해서 XAQuery 클래스의 인스턴스 생성**

```
t2101.ResFileName = "C:₩₩eBEST₩₩xingAPI₩₩Res₩₩t2101.res“
```
**# Res 파일을 앞서 생성한 클래스의 인스턴스에 등록함**

```
t2101.SetFieldData("t2101InBlock", "focode", 0, "")
```
**# 입력데이타 설정** SetFieldData의 1번째 파라미터는 블록명이고, 2번째 파라미터는 필드명. 3번째 파라미터에는 단일 데이터를 조회할 때는 0을 지정, 4번째 파라미터는 필드에 해당하는 입력.

```
t2101.Request(False) #서버에 요청
```

```
while 선물현재가.수신여부 == False: #선물현재가를 현재 수신 받지 않고 있다면
    pythoncom.PumpWaitingMessages() # 이벤트대기. While 은 반복문.
```

```
t2101.GetFieldData("t2101OutBlock", "hname", 0
```
**#서버로부터 데이터 받기. GetFieldData 메서드를 사용해 t1102OutBlock 블록명과 원하는 필드명을 입력해 데이터를 가져올 수 있습니다.**

https://wikidocs.net/3683 파이썬으로 배우는 알고리즘 트레이딩 참고

# 6-3) 선물 현재가 코드 해설(T2101 수신 부분)

```
class 선물현재가:
    수신여부 = False

    def OnReceiveData(self, code):

        선물현재가.수신여부 = True

        print("종목명 : " + t2101.GetFieldData("t2101OutBlock", "hname", 0))
        print("현재가 : " + t2101.GetFieldData("t2101OutBlock", "price", 0))
        print("전일대비구분 : " + t2101.GetFieldData("t2101OutBlock", "sign", 0))
        print("전일대비 : " + t2101.GetFieldData("t2101OutBlock", "change", 0))
        print("등락률 : " + t2101.GetFieldData("t2101OutBlock", "diff", 0))
        print("거래량 : " + t2101.GetFieldData("t2101OutBlock", "volume", 0))
        print("상한가 : " + t2101.GetFieldData("t2101OutBlock", "uplmtprice", 0))
        print("하한가 : " + t2101.GetFieldData("t2101OutBlock", "dnlmtprice", 0))
        print("시가 : " + t2101.GetFieldData("t2101OutBlock", "open", 0))
        print("고가 : " + t2101.GetFieldData("t2101OutBlock", "high", 0))
        print("저가 : " + t2101.GetFieldData("t2101OutBlock", "low", 0))
```

* OnReceiveData는 메써드로 이베스트증권 Reference 가이드에서 ReceiveData 라고 한 메써드임.
역시 파이썬이라서 앞에 On을 붙임(예 : 로그인시 Onlogin

* TR 2101 요청한 데이터가 도착했을 때 실행되는 영역. 수신확인에 필요한 변수인 수신여부를 True로 바꾸고 필요한 데이터를 전부 출력해서 확인(수신여부=False 는 현재 수신이 안되었다는 뜻)

***Outblock + 는 문자열을 연결시킬때 사용.**
**예) print (“과자”+”사탕”) =>과자사탕**
**print(“과자” , “사탕 " )으로 하면 과자 사탕 으로 나옴**

# 6-4) 선물 현재가 코드 해설 (TR 요청 부분)

```
t2101 = win32com.client.DispatchWithEvents("XA_DataSet.XAQuery", 선물현재가)
t2101.ResFileName = "C:₩₩eBEST₩₩xingAPI₩₩Res₩₩t2101.res"
t2101.SetFieldData("t2101InBlock", "focode", 0, "") # 종목코드
t2101.Request(False)
```

t2101이라는 TR을 담당하는 객체를 만듭니다.

t2101이 사용할 res파일을 할당합니다

t2101이 필요한 데이터인 종목코드를 입력해서

선물현재가의 데이터를 요청합니다

**res 파일이란?**
res 파일은 TR에 대한 정보가 들어있는 파일이다.
res 파일을 할당 받은 이후에 TR로서의 역할을 할 수 있다

**필요한 데이터는 어디에서 알 수 있나요?**
DevCenter에 해당 TR을 검색하면 요청전에 필요한 데이터인 InBlock에 해당하는 데이터를 알 수 있다

Res 파일은 DevCenter에서 사전에 '전부 다운로드 해놓아야 함!

```
while 선물현재가.수신여부 == False:
    pythoncom.PumpWaitingMessages()
```

**While 문은 반복문으로 해당 조건이 만족될 때까지 무한 반복. 여기서는 수신여부가 true로 바뀔 때까지 메세지를 기다림. TR 역시 비동기 방식이기 때문에 응답을 받을 때 까지 기다리는 코드를 사용**

# 6-5) 선물현재가.py 실행 결과

**선물현재가.py 파일이 있는 폴더 내에서 cmd실행 후**

**python 선물현재가.py <- 아래처럼 타자를 치고 엔터**

*** 주의 : 실행전 코드 내에서 본인의 아이디, 접속비번 등으로 수정해야 함. 선물 종목 코드도 수정**

```
C:\Users\cjk07\anaconda3\python.exe "C:/Users/cjk07/Desktop/Python Testing/선물현재가.py"
로그인 성공
종목명 : 코스피200 F 202006
현재가 : 267.00
전일대비구분 : 5
전일대비 : 1.30
등락률 : -0.48
거래량 : 362321
상한가 : 289.75
하한가 : 246.85
시가 : 266.95
고가 : 269.45
저가 : 265.10

Process finished with exit code 0
```

# 6-6) 로그인+선물 현재가 통합 코드 - 선물현재가(로그인 임포트).py

**로그인과 선물 현재가 코드가 분리 –로그인 정보가 담긴 로그인.py 파일을 현재 코드로 가져와서 사용하는 방식**

**실제 코딩할때는 아래처럼 로그인, 선물현재가 Class를 만들고 별도의 모듈로 분리해서, 다른 곳에서 불러오는 형태로 사용**

```
from 로그인 import *

class 선물현재가:
    수신여부 = False

    def OnReceiveData(self, code):
        선물현재가.수신여부 = True

        print("종목명 : " + t2101.GetFieldData("t2101OutBlock", "hname", 0))
        print("현재가 : " + t2101.GetFieldData("t2101OutBlock", "price", 0))
        print("전일대비구분 : " + t2101.GetFieldData("t2101OutBlock", "sign", 0))
        print("전일대비 : " + t2101.GetFieldData("t2101OutBlock", "change", 0))
        print("등락률 : " + t2101.GetFieldData("t2101OutBlock", "diff", 0))
        print("거래량 : " + t2101.GetFieldData("t2101OutBlock", "volume", 0))
        print("상한가 : " + t2101.GetFieldData("t2101OutBlock", "uplmtprice", 0))
        print("하한가 : " + t2101.GetFieldData("t2101OutBlock", "dnlmtprice", 0))
        print("시가 : " + t2101.GetFieldData("t2101OutBlock", "open", 0))
        print("고가 : " + t2101.GetFieldData("t2101OutBlock", "high", 0))
        print("저가 : " + t2101.GetFieldData("t2101OutBlock", "low", 0))

# ------------------------------------------------------------------------------
# t2101
# ------------------------------------------------------------------------------
t2101 = win32com.client.DispatchWithEvents("XA_DataSet.XAQuery", 선물현재가)

t2101.ResFileName = "C:\eBEST\xingAPI\Res\t2101.res"

t2101.SetFieldData("t2101InBlock", "focode", 0, "") # 종목코드

t2101.Request(False)

while 선물현재가.수신여부 == False:
    pythoncom.PumpWaitingMessages()
```

**from 파일명 import ***

**From은 파일을 지정할때 쓰인다**

*** 는 전부라는 의미이다**

**다른 파일들도 (로그인 임포트).py 는 로그인 정보를 import 해서 불러오는 형식으로 코딩된 파일임**

# 이베스트증권 XingAPI

# 초보자용 Python 개발가이드

# 제3장 선물 실시간 시세받기 코딩

**XA REAL 이용, FC0 , 해당 실시간TR /REAL 목록은 DevCenter에서 확인**

**실시간으로 선물 현재가를 불러올 때 사용(한번 요청하면 시세가 장종료까지 계속 들어옴)**
**(선물 현재가 TR은 한번 요청하면 그 순간의 현재가만 결과값으로 한번 오고 끝)**

# 7-1) 선물 실시간 코드 (선물실시간.py)

```
import win32com.client
import pythoncom

class 로그인:
    로그인여부 = False

    def OnLogin(self, code, msg):
        if code == "0000":
            print("로그인 성공")
            로그인.로그인여부 = True
        else:
            print("로그인 실패")



class 선물실시간:
    수신여부 = False

    def OnReceiveRealData(self, code):
        print("체결시간 : " + FC0.GetFieldData("OutBlock", "chetime"))
        print("전일대비구분 : " + FC0.GetFieldData("OutBlock", "sign"))
        print("전일대비 : " + FC0.GetFieldData("OutBlock", "change"))
        print("등락율 : " + FC0.GetFieldData("OutBlock", "drate"))
        print("현재가 : " + FC0.GetFieldData("OutBlock", "price"))
        print("시가 : " + FC0.GetFieldData("OutBlock", "open"))
        print("고가 : " + FC0.GetFieldData("OutBlock", "high"))
        print("저가 : " + FC0.GetFieldData("OutBlock", "low"))
        print("체결구분 : " + FC0.GetFieldData("OutBlock", "cgubun"))
        print("체결량 : " + FC0.GetFieldData("OutBlock", "cvolume"))
        print("누적거래량 : " + FC0.GetFieldData("OutBlock", "volume"))
        print("누적거래대금 : " + FC0.GetFieldData("OutBlock", "value"))
        print("체결강도 : " + FC0.GetFieldData("OutBlock", "cpower"))
        print("매도호가1 : " + FC0.GetFieldData("OutBlock", "offerho1"))
        print("매수호가1 : " + FC0.GetFieldData("OutBlock", "bidho1"))
        print("미결제약정수량 : " + FC0.GetFieldData("OutBlock", "openyak"))
        print("KOSPI200지수 : " + FC0.GetFieldData("OutBlock", "k200jisu"))
        print("장운영정보 : " + FC0.GetFieldData("OutBlock", "jgubun"))
        print("단축코드 : " + FC0.GetFieldData("OutBlock", "futcode"))
```

```
# ------------------------------------------------------------------------------
# login
# ------------------------------------------------------------------------------
세션 = win32com.client.DispatchWithEvents("XA_Session.XASession",
로그인)

아이디 = "" # 내 아이디
비밀번호 = "" # 내 비밀번호
공인인증번호 = "" # 내 공인인증번호

세션.ConnectServer("demo.ebestsec.co.kr", 20001)
세션.Login(아이디, 비밀번호, 공인인증번호, 0, 0)

while 로그인.로그인여부 == False:
    pythoncom.PumpWaitingMessages()

# ------------------------------------------------------------------------------
# FC0
# ------------------------------------------------------------------------------
FC0 = win32com.client.DispatchWithEvents("XA_DataSet.XAReal",
선물실시간)

FC0.ResFileName = "C:₩₩eBEST₩₩xingAPI₩₩Res₩₩FC0.res"
FC0.SetFieldData("InBlock", "futcode", "") # 종목코드
FC0.AdviseRealData()

while 선물실시간.수신여부 == False:
    pythoncom.PumpWaitingMessages()
```

# 7-2) 선물실시간 코드 해설(**선물실시간.py**)

```
class 선물실시간:
    수신여부 = False  # False면 수신이 안되어 있다는 뜻

    def OnReceiveRealData(self, code):
        print("체결시간 : " + FC0.GetFieldData("OutBlock", "chetime"))
        print("전일대비구분 : " + FC0.GetFieldData("OutBlock", "sign"))
        print("전일대비 : " + FC0.GetFieldData("OutBlock", "change"))
        print("등락율 : " + FC0.GetFieldData("OutBlock", "drate"))
        print("현재가 : " + FC0.GetFieldData("OutBlock", "price"))
        print("시가 : " + FC0.GetFieldData("OutBlock", "open"))
        print("고가 : " + FC0.GetFieldData("OutBlock", "high"))
        print("저가 : " + FC0.GetFieldData("OutBlock", "low"))
        print("체결구분 : " + FC0.GetFieldData("OutBlock", "cgubun"))
        print("체결량 : " + FC0.GetFieldData("OutBlock", "cvolume"))
        print("누적거래량 : " + FC0.GetFieldData("OutBlock", "volume"))
        print("누적거래대금 : " + FC0.GetFieldData("OutBlock", "value"))
        print("체결강도 : " + FC0.GetFieldData("OutBlock", "cpower"))
        print("매도호가1 : " + FC0.GetFieldData("OutBlock", "offerho1"))
        print("매수호가1 : " + FC0.GetFieldData("OutBlock", "bidho1"))
        print("미결제약정수량 : " + FC0.GetFieldData("OutBlock", "openyak"))
        print("KOSPI200지수 : " + FC0.GetFieldData("OutBlock", "k200jisu"))
        print("장운영정보 : " + FC0.GetFieldData("OutBlock", "jgubun"))
        print("단축코드 : " + FC0.GetFieldData("OutBlock", "futcode"))
```

OnReceiveData는 요청한 데이터가 도착했을 때 실행됩니다
주문가격과 주문 수량을 출력해서 정상적으로 주문이 완료되었는지 확인합니다
+ 는 문자열끼리 합칠 때 사용(앞에서 설명함)

# 7-3) 선물 실시간 코드 해설

```
FC0 = win32com.client.DispatchWithEvents("XA_DataSet.XAReal", 선물실시간)

FC0.ResFileName = "C:₩₩eBEST₩₩xingAPI₩₩Res₩₩FC0.res"
FC0.SetFieldData("InBlock", "futcode", "") # 종목코드
FC0.AdviseRealData()
```

FC0 이라는 REAL을 담당하는 객체를 만듭니다.
FC0 이 사용할 res파일을 할당합니다
FC0 이 필요한 데이터를 입력해서
선물현재가의 데이터를 요청합니다

*** 기본적으로 선물실시간 Real 은 선물현재가 TR와 동일한 구조로 코딩한다. 차이점은**

**1) TR**에서는 데이터 요청시 t2101.Request(False) 를 사용하며, **Real** 에서는 데이터 요청시
FC0.AdviseRealData() 로 사용

**2) TR에서는 수신했나 false 일 때 선물현재가.수신했나= True 가 더 들어감(REAL 에서는 안들어 감)**

```
while 선물실시간.수신여부 == False:
    pythoncom.PumpWaitingMessages()
```

# 7-4) pythoncom.PumpWaitingMessages() 관련

```
while 선물실시간.수신여부 == False:
    pythoncom.PumpWaitingMessages()
```

위 코드는 아래 코드로 대체할 수 있다.

```
While True: #장종료조건추가
    pythoncom.PumpWaitingMessages()
```

** PumpWaitingMessages 관련 다른 의견 의견 : real은 PumpWaitingMessgaes 하지 않는게 맞을거에요 PumpWaitingMessages()를 사용하게되면, 무한루프를 돌게되는 구조라서 CPU리소스를 많이 잡아 먹습니다. 대신 PyQt에 있는 이벤트 루프를 활용하는게 훨씬 유리합니다

수신여부(수신했나)가 False라면 프로그램이 종료되지 않고 응답이 오기를 기다립니다. 비동기 방식이기 때문에 응답을 받을 때 까지 기다리는 코드를 사용합니다. OnReceiveRealData 하고 OnReceiveData는 PumpWaitingMessages()함수가 호출되어져야 Python COM에서 응답을 받을 수 있게 됩니다.

**선물 실시간 코드에서는 선물 현재가 코드처럼 수신여부(수신했나)를 True로 바꾸지 않는 이유**

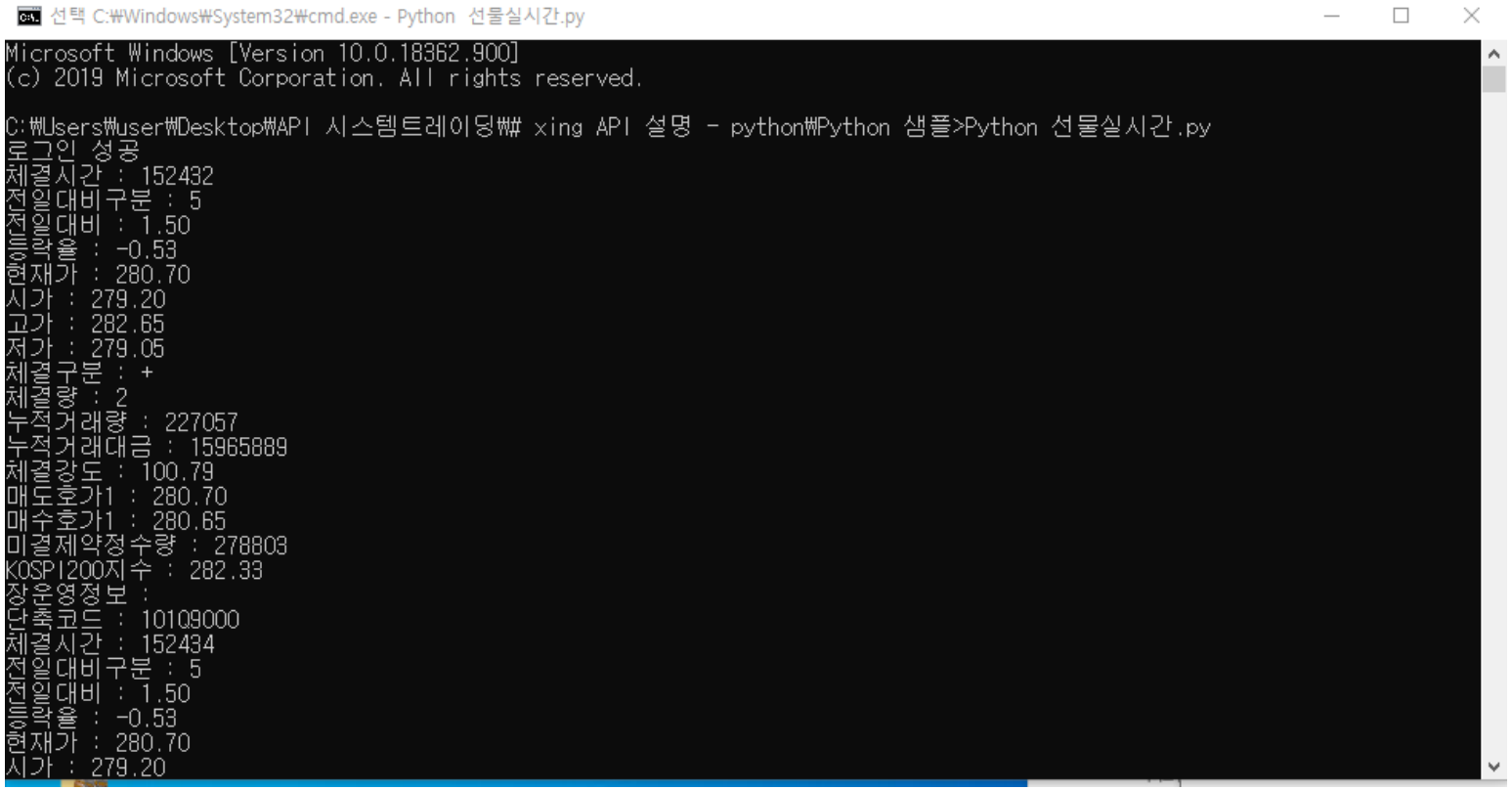
만약 수신여부가 True로 바뀐다면 더 이상 데이터를 기다리지 않고 코드가 끝나버립니다. 데이터를 하나가 아니라 지속적으로 받아와야 하는 Real에 특성상  자기의 필요에 따라 언제 수신여부를 True로 바꿔 데이터를 그만 받을지 선택해야합니다. 다만, True로 바꾸지 않기에 프로그램을 켜 놓는 동안 루프가 무한으로 돌게 되어 CPU에 부담이 커집니다.

# 7-5) 선물실시간.py 실행 결과

**선물실시간.py 파일이 있는 폴더 내에서 cmd실행 후**

**python 선물실시간.py <- 아래처럼 타자를 치고 엔터**

*** 주의 : 실행전 코드 내에서 본인의 아이디, 접속비번 등으로 수정해야 함. 선물 종목 코드도 수정**



# 이베스트증권 XingAPI

# 초보자용 Python 개발가이드

# 제4장 선물옵션 매수/매도주문 코딩

- XAquery 이용, CFOAT00100 선물옵션 정상주문(해당 TR은 DevCenter에서 확인)

- 선물옵션 관련 매수/매도 주문을 할 때 사용
  (선물옵션 정정주문은 CFOAT00200 사용, 취소주문은 CFOAT00300 사용)

# 8-1) 선물주문 코드(선물주문.py)

```
import win32com.client
import pythoncom

class 로그인:
    로그인여부 = False

    def OnLogin(self, code, msg):
        if code == "0000":
            print("로그인 성공")
            로그인.로그인여부 = True
        else:
            print("로그인 실패")



class 선물주문:
    수신여부 = False

    def OnReceiveData(self, code):
        선물주문.수신여부 = True

        print("주문가격 : " + CFOAT00100.GetFieldData("CFOAT00100OutBlock1", "OrdPrc", 0))
        print("주문수량 : " + CFOAT00100.GetFieldData("CFOAT00100OutBlock1", "OrdQty", 0))
        print("주문번호 : " + CFOAT00100.GetFieldData("CFOAT00100OutBlock2", "OrdNo", 0))


# ------------------------------------------------------------------------------
# login
# ------------------------------------------------------------------------------

세션 = win32com.client.DispatchWithEvents("XA_Session.XASession", 로그인)

아이디 = "" # 내 아이디
비밀번호 = "" # 내 비밀번호
공인인증번호 = "" # 내 공인인증번호

세션.ConnectServer("demo.ebestsec.co.kr", 20001)
세션.Login(아이디, 비밀번호, 공인인증번호, 0, 0)

while 로그인.로그인여부 == False:
    pythoncom.PumpWaitingMessages()
```

```
# ------------------------------------------------------------------------------
# CFOAT00100
# ------------------------------------------------------------------------------
CFOAT00100 = win32com.client.DispatchWithEvents("XA_DataSet.XAQuery",
선물주문)

CFOAT00100.ResFileName = "C:\eBEST\xingAPI\Res\CFOAT00100.res"

CFOAT00100.SetFieldData("CFOAT00100InBlock1", "AcntNo", 0, "") # 계좌번호
CFOAT00100.SetFieldData("CFOAT00100InBlock1", "Pwd", 0, "") # 계좌비밀번호
CFOAT00100.SetFieldData("CFOAT00100InBlock1", "FnoIsuNo", 0, "") # 종목코드
CFOAT00100.SetFieldData("CFOAT00100InBlock1", "BnsTpCode", 0, "") #매매구분

# 매매구분
# 1 - 매도 , 2 - 매수

CFOAT00100.SetFieldData("CFOAT00100InBlock1", "FnoOrdprcPtnCode", 0,
"")

# 호가코드
# 00 - 지정가 , 03 - 시장가 , 05 – 조건부지정가, 06 - 최유리지정가
# 10 - 지정가(IOC) , 20 - 지정가(FOK), 13 - 시장가(IOC), 23 - 시장가(FOK)
# 16 - 최유리지정가(IOC)

CFOAT00100.SetFieldData("CFOAT00100InBlock1", "OrdPrc", 0, "250") # 주문가격
CFOAT00100.SetFieldData("CFOAT00100InBlock1", "OrdQty", 0, "1") # 주문수량

CFOAT00100.Request(False)

while 선물주문.수신여부 == False:
    pythoncom.PumpWaitingMessages()
```

# 8-3) 선물 주문 코드 해설(TR 코드 방식과 동일)

```
CFOAT00100 = win32com.client.DispatchWithEvents("XA_DataSet.XAQuery", 선물주문)

CFOAT00100.ResFileName = "C:₩₩eBEST₩₩xingAPI₩₩Res₩₩CFOAT00100.res"

CFOAT00100.SetFieldData("CFOAT00100InBlock1", "AcntNo", 0, "") # 계좌번호
CFOAT00100.SetFieldData("CFOAT00100InBlock1", "Pwd", 0, "") # 계좌비밀번호
CFOAT00100.SetFieldData("CFOAT00100InBlock1", "FnoIsuNo", 0, "") # 종목코드
CFOAT00100.SetFieldData("CFOAT00100InBlock1", "BnsTpCode", 0, "")

# 매매구분
# 1 - 매도
# 2 - 매수

CFOAT00100.SetFieldData("CFOAT00100InBlock1", "FnoOrdprcPtnCode", 0, "")

# 호가코드
# 00 - 지정가
# 03 - 시장가
# 05 - 조건부지정가
# 06 - 최유리지정가
# 10 - 지정가(IOC)
# 20 - 지정가(FOK)
# 13 - 시장가(IOC)
# 23 - 시장가(FOK)
# 16 - 최유리지정가(IOC)

CFOAT00100.SetFieldData("CFOAT00100InBlock1", "OrdPrc", 0, "") # 주문가격
CFOAT00100.SetFieldData("CFOAT00100InBlock1", "OrdQty", 0, "") # 주문수량

CFOAT00100.Request(False)
```

CFOAT00100 라는 TR을 담당하는 객체를 만듭니다.
CFOAT00100가 사용할 res파일을 할당합니다
CFOAT00100가 필요한 데이터를 입력해서
선물현재가의 데이터를 요청합니다

**호가코드, 매매구분에 대한 정보는 어디서 얻을 수 있나요?**
DevCenter에 해당 부분을 클릭하면 Field 속성에 정보가 나옵니다

# 8-2) 선물 주문 코드 해설 및 실행 결과

```
class 선물주문:
    수신여부 = False

    def OnReceiveData(self, code):
        선물주문.수신여부 = True
```

OnReceiveData는 요청한 데이터가 도착했을 때 실행. 수신확인에 필요한 변수인 수신여부를 True로 바꾸고 주문가격과 주문수량, 주문번호을 출력해서 정상적으로 주문이 완료되었는지 확인

```
        print("주문가격 : " + CFOAT00100.GetFieldData("CFOAT00100OutBlock1", "OrdPrc", 0))
        print("주문수량 : " + CFOAT00100.GetFieldData("CFOAT00100OutBlock1", "OrdQty", 0))

        print("주문번호 : " + CFOAT00100.GetFieldData("CFOAT00100OutBlock2", "OrdNo", 0))
```

**실행 전 코드에서 아이디, 접속 비번 등 변경하고, 종목코드도 넣고 주문수량과 주문가격도 아래처럼 변경해야 함**

```
CFOAT00100.SetFieldData("CFOAT00100InBlock1", "FnoIsuNo", 0, "") # 종목코드
CFOAT00100.SetFieldData("CFOAT00100InBlock1", "OrdPrc", 0, "250") # 주문가격
CFOAT00100.SetFieldData("CFOAT00100InBlock1", "OrdQty", 0, "1") # 주문수량
```

**** 선물주문.py 파일이 있는 폴더에서 cmd 실행 -> python 선물주문.py ->실행하면 아래와 같은 메시지가 뜨면 성공(장중에만 테스트 가능)**

# 8-3) 선물주문 코드 – 선물주문(로그인 임포트).py

**로그인과 선물 주문 코드가 분리 –로그인 정보가 담긴 login.py 파일을 현재 코드로 가져와서 사용하는 방식**

```
from 로그인 import *

class 선물주문:
    수신여부 = False

    def OnReceiveData(self, code):
        선물주문.수신여부 = True

        print("주문가격 : " + CFOAT00100.GetFieldData("CFOAT00100OutBlock1", "OrdPrc", 0))
        print("주문수량 : " + CFOAT00100.GetFieldData("CFOAT00100OutBlock1", "OrdQty", 0))
        print("주문번호 : " + CFOAT00100.GetFieldData("CFOAT00100OutBlock2", "OrdNo", 0))


# ------------------------------------------------------------------------------
# CFOAT00100
# ------------------------------------------------------------------------------
CFOAT00100 = win32com.client.DispatchWithEvents("XA_DataSet.XAQuery",
선물주문)

CFOAT00100.ResFileName = "C:₩₩eBEST₩₩xingAPI₩₩Res₩₩CFOAT00100.res"

CFOAT00100.SetFieldData("CFOAT00100InBlock1", "AcntNo", 0, "") # 계좌번호
CFOAT00100.SetFieldData("CFOAT00100InBlock1", "Pwd", 0, "") # 계좌비밀번호
CFOAT00100.SetFieldData("CFOAT00100InBlock1", "FnoIsuNo", 0, "") # 종목코드
CFOAT00100.SetFieldData("CFOAT00100InBlock1", "BnsTpCode", 0, "")

# 매매구분
# 1 - 매도
# 2 - 매수
CFOAT00100.SetFieldData("CFOAT00100InBlock1", "FnoOrdprcPtnCode", 0, "")

# 호가코드
# 00 - 지정가
# 03 - 시장가
# 05 - 조건부지정가
# 06 - 최유리지정가
# 10 - 지정가(IOC)
# 20 - 지정가(FOK)
# 13 - 시장가(IOC)
# 23 - 시장가(FOK)
# 16 - 최유리지정가(IOC)

CFOAT00100.SetFieldData("CFOAT00100InBlock1", "OrdPrc", 0, "") # 주문가격
CFOAT00100.SetFieldData("CFOAT00100InBlock1", "OrdQty", 0, "") # 주문수량

CFOAT00100.Request(False)

while 선물주문.수신여부 == False:
    pythoncom.PumpWaitingMessages()
```

# 9) PyQt 개요 및 PyQt 설치방법

**PyQt 개요**

다른 프로그래밍 언어들과 마찬가지로 Python 역시 여러 툴들을 이용하여 GUI를 만들고 이용할 수 있다.

현재 가장 폭넓게 쓰이고 있는 GUI툴은 PyQt라는 GUI 툴이다

Qt designer이라는 프로그램을 통하여 쉽게 GUI를 디자인 가능하다

아나콘다를 설치하게 되면 기본적으로 깔리게 되는 툴이고 간편하게 쓸 수 있어 많은 수의 Python 트레이더들이 사용하고 있다.

**PyQt 설치방법**

아나콘다를 설치한 이후(이미 되어 있다면 생략)

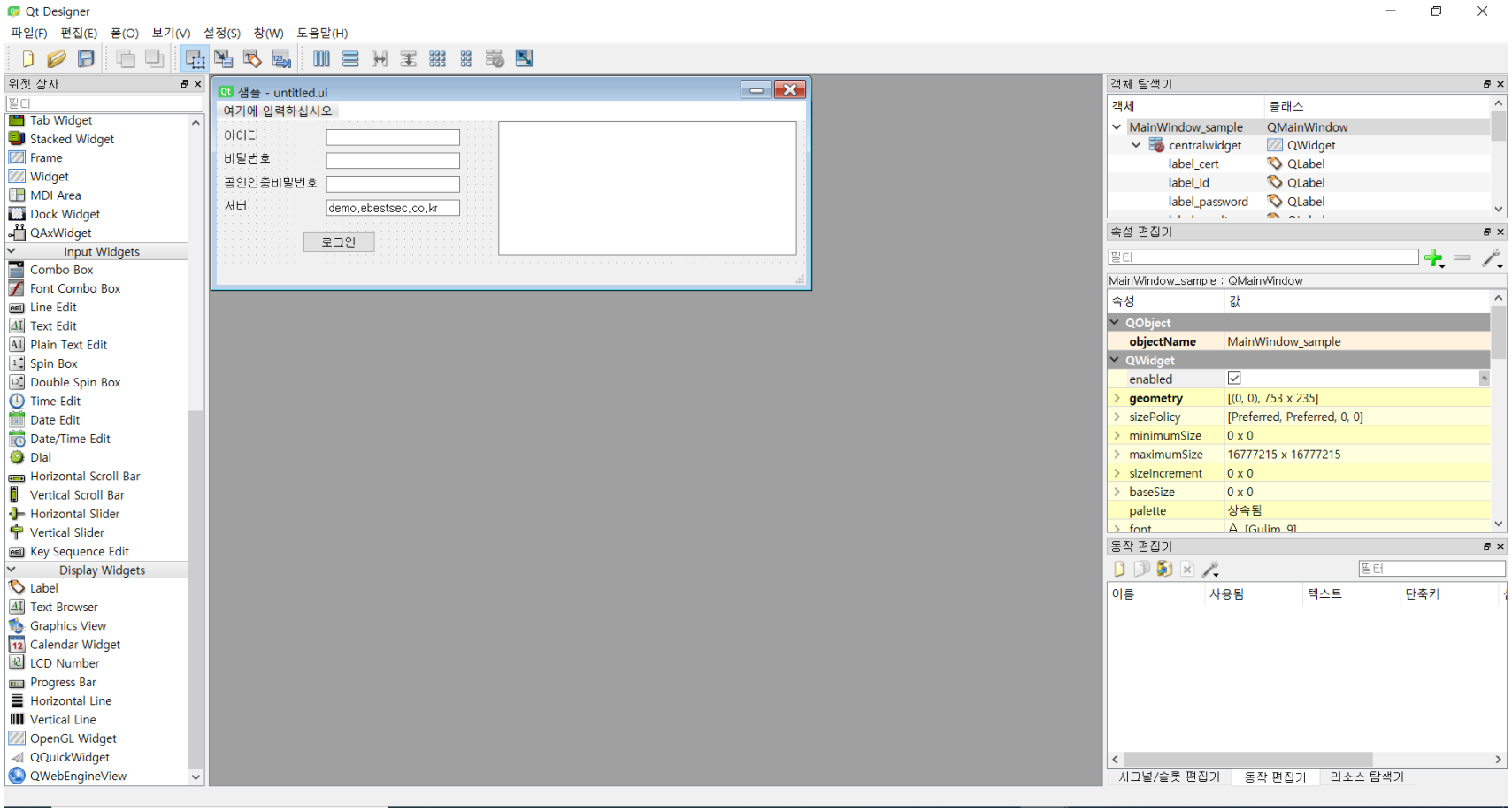
Anaconda Prompt를 실행

명령어를 차례로 입력

```
python –m pip install –upgrade pip
pip install pyqt5
pip install pyqt5-tools
```

C:₩ProgramData₩Anaconda3₩Library₩bin에 들어가 designer을 실행

# 9-1) PyQt designer




드래그 & 드랍으로 위젯들을 넣어 배치할 수 있다
저장을 하게 되면 .ui 라는 파일로 저장이 된다

# 9-2) PyQt 코드와 대략적 설명 (PyQtSample.py)

```
import sys
import win32com.client
import pythoncom
from PyQt5.QtWidgets import *
from PyQt5 import uic

form = uic.loadUiType("untitled.ui")[0]
```

.ui 파일을 form이라는 변수에 저장한다
*현재 코드(.py)와 designer에서 저장한 파일(.ui)이 같은 폴더에 있어야 한다

```
class 로그인:
    로그인여부 = False

    def OnLogin(self, szCode, szMsg):
        window.textBrowser_output.append("szCode " + szCode + ", szMsg " + szMsg)
        if szCode == "0000":
            window.textBrowser_output.append("로그인 성공")
            로그인.로그인여부 = True
        else:
            window.textBrowser_output.append("로그인 실패")

class MainWindow(QMainWindow, form):
    def __init__(self):
        super().__init__()
        self.setupUi(self)
        self.pushButton_login.clicked.connect(self.login_clicked)
```

변수 form과 PyQt의 코드를 이용해 메인 윈도우를 설정한다
*현 코드에서는 UI와 과 로그인 버튼의 이벤트를 설정 함

```
    def login_clicked(self):
        세션 = win32com.client.DispatchWithEvents("XA_Session.XASession", 로그인)

        아이디 = self.lineEdit_id.text()  # 내 아이디
        비밀번호 = self.lineEdit_password.text()  # 내 비밀번호
        공인인증번호 = self.lineEdit_cert.text()  # 내 공인인증번호

        세션.ConnectServer(self.lineEdit_server.text(), 20001)
        세션.Login(아이디, 비밀번호, 공인인증번호, 0, 0)
        while 로그인.로그인여부 == False:
            pythoncom.PumpWaitingMessages()
```

로그인 버튼이 클릭되었을 때 실행할 이벤트 함수

```
if __name__ == '__main__':
    app = QApplication(sys.argv)
    window = MainWindow()
    window.show()
    app.exec_()
```

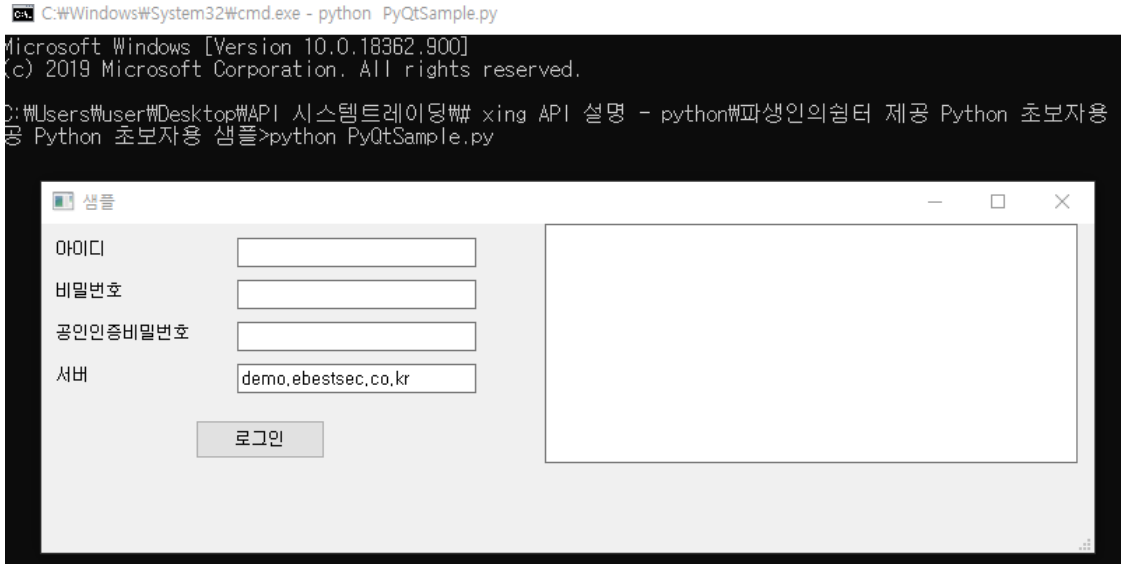
윈도우 창을 띄우기 위해 필요한 코드

*어떻게 작동되는지 궁금하다면 샘플 파일을 참고

# 9-3) PyQtSample.py 실행 결과(cmd와 파이참)

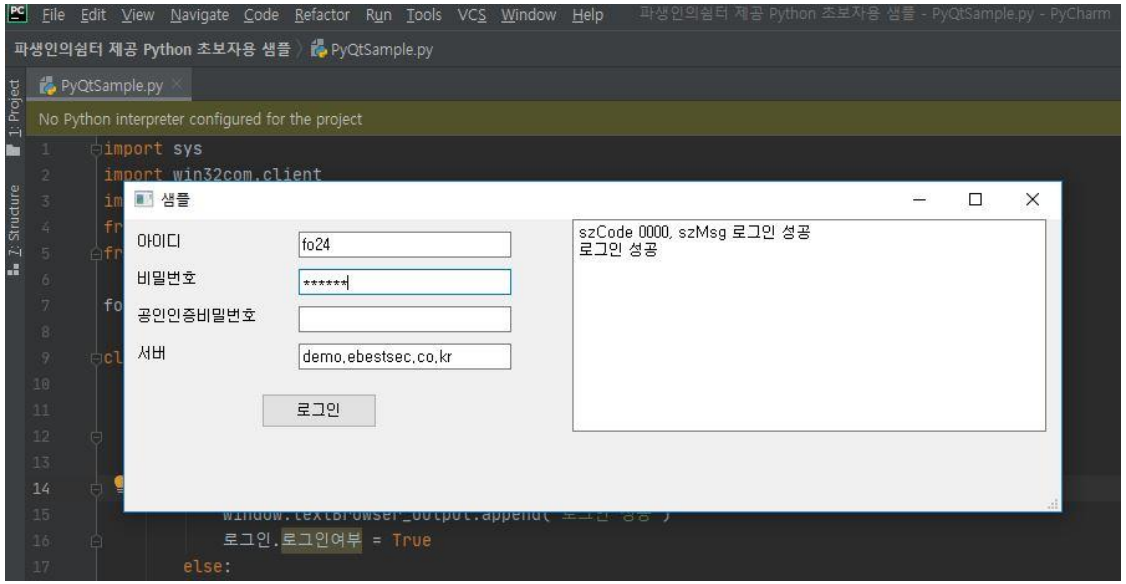
1) Cmd에서 실행 -> PyQtSample.py 파일이 있는 폴더 내에서 cmd실행 후

python PyQtSample.py <- 아래처럼 타자를 치고 엔터하면 오른쪽 처럼 form이 열림 -> 여기서 아이디, 비밀번호 등을 치고 로그인 클릭하면 오른쪽에 성공여부 메세지가 뜸




2) Pycharm에서 실행 -> PyQtSample.py 를 프로젝트로 불러옴 -> 메뉴 settings에서 Project Interpreter에서 PyQT5를 설치-> 파일 이름에 대고 오른쪽 마우스 클릭 -> run -> 오른쪽과 같은 form이 열림

여기서 아이디, 비밀번호 등을 치고 로그인 클릭하면 오른쪽에 성공여부 메세지가 뜸



# Python으로 XingAPI 코딩할 때 유용한 사이트

**1) https://wikidocs.net/book/110 파이썬으로 배우는 알고리즘 트레이딩**

**파이썬을 기반으로 이베스트증권, 키움증권 등 API와 파이썬 코드 설명(초보자용)**

**2) https://thinkalgo.tistory.com/** <- 씽크알고(이베스트증권에서 공식적으로 제공하는 API 사이트, 씽크풀에서 운영)

**3) https://malchooni.name/** <- **말춘이의 블로그 github 등에도 관련 코드을 올리는 등 활발한 활동(중급 이상)**

**4) http://cafe.naver.com/pystock** <- **파이스탁, Python 언어로 API 코딩하는 카페 모임**

**5) https://freeprog.tistory.com/** <- **취미로 하는 프로그래밍 (이베스트증권API, 키움증권 API, Python 관련 자료)**

**6) 파생인의쉼터 http://cafe.naver.com/fo24** <- **이베스트증권 등 VBA, C#, Python 샘플 제공, 국내, 해외 파생 및 선물옵션 국내 최저 협의수수료 제공**