서문

QuantConnect 란?

QuantConnect?

QuantConnect의 세 가지 핵심 플랫폼

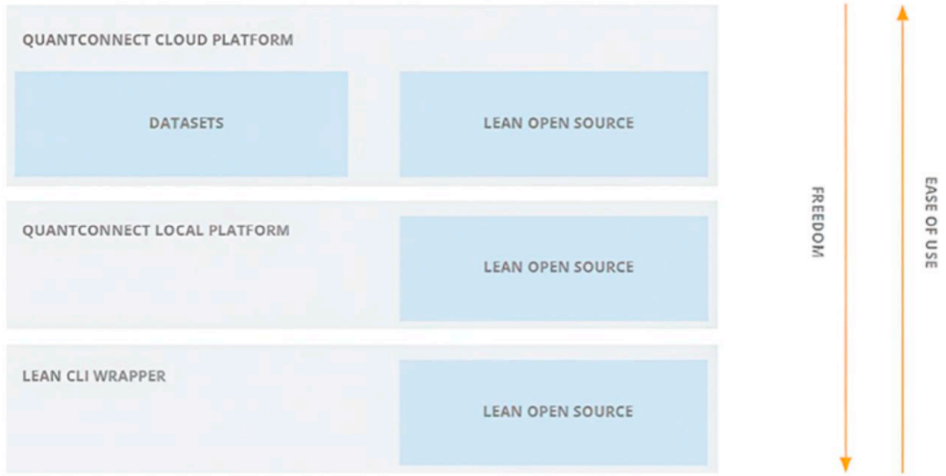
구성 요소	설명
클라우드 플랫폼	백테스트, 최적화, 라이브 트레이딩까지 클라우드에서 자동 처리. 팀 온보딩, 인프라 구성 자동화
로컬 플랫폼 (온프레미스)	독점 데이터 사용 가능, 사용자 정의 라이브러리 설치 가능. 조직 내 서버에서 독립 운영
LEAN CLI	터미널 기반 도구. 백테스트, 최적화, 실시간 실행 모두 가능. 결과는 JSON으로 출력

연구 환경 (Research)

- **Jupyter 기반 환경 제공** (클라우드)
- 실시간으로 연결된 **기본/대체/금융 데이터셋**
- **FIGI, ISIN, CUSIP 등**으로 데이터 태깅됨
- 머신러닝 라이브러리 (sklearn, xgboost 등) 사용 가능

백테스트 (Backtest)

- 슬리피지, 수수료, 마진 모델 등 **현실적인 조건 반영**
- **초고속 실행** 가능 (틱 단위 해상도 포함)
- 사용자 정의 데이터 연동 가능
- 일 20,000개 이상 백테스트 실행



QuantConnect?

최적화 (Optimization)

- **수천 개의 백테스트 병렬 실행**
- 히트맵 기반 **민감도 분석**
- 각 백테스트에 대한 **개별 로그/거래 이력 확인** 가능

라이브 트레이딩 (Live Trading)

- 공동 배치 환경에서 실시간 거래 가능
- **450억 달러 이상**의 월 거래량 처리
- 1,200개 유동성 공급자 연결 지원

기술 기반: LEAN 엔진

- QuantConnect는 **오픈소스 알고리즘 거래 엔진인 LEAN** 위에서 작동
- C# 기반 데이터 처리 + Python 연동
- 개인 서버에서 자유롭게 실행 가능 (상업적 라이선스 허용)
- **LEAN CLI**를 통해 글로벌 환경에서 연구–실행 전 주기 지원

이 책의 대상 독자층

- **금융/투자에 관심 있는 대학생, 트레이더, 연금/헤지펀드 매니저**
- **AI·알고리즘 거래 전략 강화**에 관심 있는 투자사
- Python(pandas, NumPy, scikit-learn)과 금융지식 보유자
- 전략 변경에 따른 결과가 **왜, 언제 바뀌는지** 궁금한 사람

제1장 자본 시장의 기초

현대 금융 시장의 핵심 개념과 이러한 개념이 QuantConnect에서 어떻게 표현되는지 소개합니다.

현대 미국 시장, 데이터 피드, 그리고 이후 장에서 사용될 자산 클래스에 대해 다룰 것입니다.

자본 시장의 기초 - 시장 메커니즘

1. **미국 주요 거래소**
   - **미국에는 11개의 주요 증권 거래소가 있음.**
   - **가장 큰 두 곳은 NYSE(뉴욕 증권 거래소)와 NASDAQ(전국 증권 딜러 협회 자동 호가 시스템)**
2. **단일 데이터 피드와 NBBO**
   - 모든 거래는 **증권 정보 처리기(SIP)**를 통해 통합됨.
   - SIP는 NBBO(National Best Bid and Offer)를 계산함.
     - NBBO: 미국 전역에서 가장 좋은 매수/매도 호가
     - 100주 이상의 호가만 NBBO 계산에 포함됨
3. 시장 외부(order off-exchange) 체결
   - **중개업체는 마켓 메이커에게 주문을 넘겨 장외에서 체결할 수 있음.**
   - 마켓 메이커는 NBBO 범위 내에서 체결 가격을 제시해야 함.
   - 장외 체결된 거래는 거래 보고 시설(TRF)에 보고되고 SIP에 포함됨
4. DMA (Direct Market Access)
   - **일부 중개업체는 직접 거래소 접근(DMA) 기능을 제공.**
   - DMA를 사용하면 주문이 특정 거래소로 바로 전달되지만,
     - **NBBO보다 불리한 가격에 체결될 위험이 있어 주의 필요**

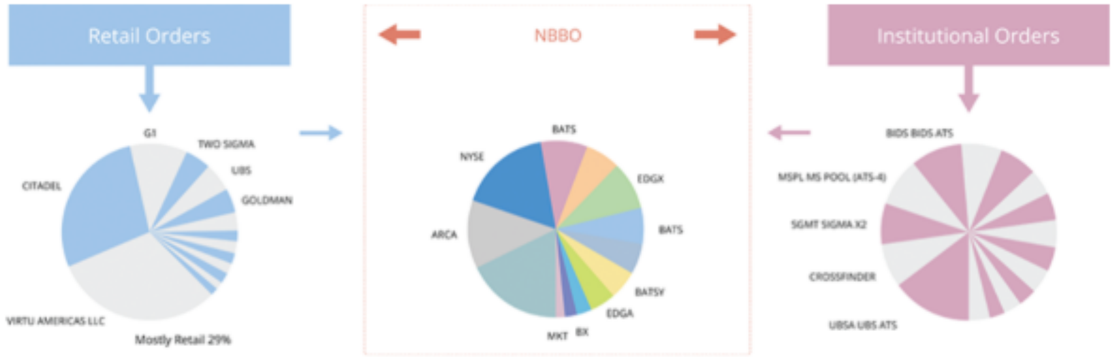


Figure 1.1 Flow of retail and institutional traffic across public and private markets, and the origin of national best pricing.

자본 시장의 기초 - 시장 참여자(1)

1. 거래는 연극이다

- 거래는 마치 극장에서 벌어지는 연극처럼 정해진 무대와 역할, 규칙이 있는 고도로 조율된 활동이다.
- 이 무대는 지정가 주문장(limit order book).

2. 지정가 주문장의 구조

- 지정가 주문장은 매수호가*(bid)*와 매도호가*(ask)*를 나열한 원장.
- 매수호가 < 매도호가 → 이 둘 사이에는 스프레드(spread)가 존재.
- 시장가 주문(market order)은 바로 체결되지만, 크기가 클 경우 주문장을 따라 올라가며 가격에 영향을 줄 수 있음 ("북 워킹- walk-the-book”)

3. 시장 참여자 = 배우들

유동성 트레이더	마켓 메이커	정보를 가진 트레이더
• 우월한 정보(항상 내부자 정보는 아님)로부터 이익을 얻는 것 이외의 목적으로 포지션에 들어가거나 나오는 것이 주요 목표인 트레이더 • 예: 포트폴리오 조정, 단순 매매 등 • 때로는 "노이즈 트레이더"로 불리기도 함	• **유동성을 공급**하는 역할. • 매수호가와 매도호가 사이에 주문을 내고 **스프레드 차익으로 수익**을 얻음. • **지정가 주문에 리베이트 제공** 등으로 유도됨 • 전통적 플로어에서 → 현재는 **고빈도 알고리즘 기반 전자 시장으로 전환**됨.	• **일반에게 알려지지 않은 정보**로 매매하는 트레이더. • 예: 블록 거래 정보, 내부자 정보 등. • 마켓 메이커는 이들과의 거래를 **위험 요소로 간주**하고 전략을 조정함.

자본 시장의 기초 - 시장 참여자(2)

4. AI 배우의 구함!

- 인간은 이제 시장 데이터 속 패턴을 **충분히 빠르게 인식하고 대응하기 어려움**.
- 그래서 **AI와 머신러닝**이 정보를 가진 트레이더의 패턴을 식별하고, **거래 신호를 자동 탐지**하는 역할을 수행.
- 현대 시장에서 **AI는 거래 전략의 핵심 도구**가 됨

자본 시장의 기초 - 데이터와 데이터 피드

시장 데이터의 흐름

- 트레이더의 주문은 실시간 이벤트로 기록되며, 거래량은 개장/마감 시간에 급증함
- 1960년대까지는 ‘티커 테이프’로 물리적 기록, 현재는 디지털 피드로 전환됨

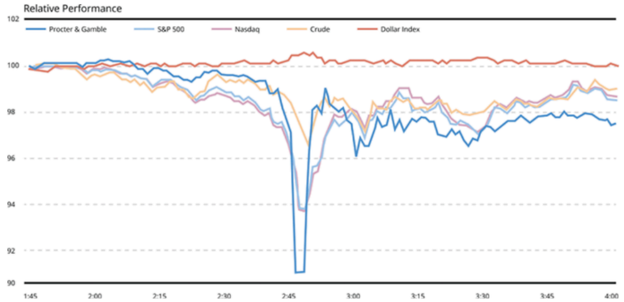


Figure 1.2 Relative performance of various assets during the market crash on May 6, 2010.

시장 이상 및 리스크

- 데이터 지연이나 오류 가능성 존재 → 예: 지연된 거래 보고
- 대표적 사례: 2010년 플래시 크래시 → 서킷 브레이커 도입

데이터 종류

- **거래 틱**: 실제 거래 발생 보고 (price, size 등 포함)
- **호가 틱**: 매수/매도 의사 표현 (bid/ask), 스프레드 형성
- **통합 바 데이터**: 틱 데이터를 일정 시간 간격으로 집계 (OHLC 형식)

QuantConnect에서, 이 데이터는 각 데이터 이벤트에서 틱 목록으로 전달됩니다. 참고로, "tick_type"은 데이터 포인트가 거래 틱인지 호가 틱인지를 식별하는 속성입니다.

```
def on_data(self, slice):
    ticks = slice.ticks.get(self._symbol, []) # 찾지 못하면 빈 배열
    for tick in ticks:
        price = tick.price # 가격 제한 또는 거래
    trade_or_quote = tick.tick_type  # TickType.TRADE 또는 QUOTE
```

QuantConnect 예제 코드

- slice.ticks, slice.bars, slice.quote_bars 등으로 실시간 데이터 수신 및 처리 가능

자본 시장의 기초 - 커스텀 및 대체 데이터

- **대체 데이터의 중요성**
  - 가격 외의 요소 (뉴스, 위치, 감성 등)도 시장 예측에 유용
  - 예: 스트리밍 뉴스로부터 Apple 관련 기사 수집 → 투자 신호로 활용 가능

- **QuantConnect 활용**
  - 60개 이상의 데이터셋을 지원하며 코드 몇 줄로 쉽게 전략에 연동 가능

    QuantConnect는 60개 이상의 데이터셋(qnt.co/book-datasets)을 호스팅하며 정기적으로 새로운 데이터셋을 온보딩합니다.

    데이터는 몇 줄의 코드로 전략에 추가할 수 있습니다. 다음 예제는 스트리밍 뉴스 서비스인 TiingoNews에서 Apple에 관한 뉴스 기사를 제공합니다:

    ```
    self._aapl = self.add_equity("AAPL", Resolution.MINUTE).symbol
    self._dataset_symbol = self.add_data(TiingoNews, self._aapl).symbol
    ```

  - 직접 정의한 커스텀 데이터셋도 add_data()로 추가 가능

    . QuantConnect에 필요한 데이터가 없다면, 데이터 형식을 클래스로 정의하고 add_data 메서드를 사용하여 전략에 추가함으로써 커스텀 데이터셋을 업로드할 수 있습니다. 다음 코드 스니펫에서, 우리는 데이터의 속성을 정의하고 소스 CSV 파일을 파싱할 커스텀 클래스인 MyFactorDataset을 전달하고 있습니다:

    ```
    self._custom_symbol = self.add_data(MyFactorDataset, "Factors").symbol
    ```

자본 시장의 기초 - 브로커리지와 거래 비용

브로커(Broker): 자본시장과 투자자 사이의 **중개자** 역할

- 자산 보유, 거래 청산 및 결제
- **SEC/FINRA 마진 규칙 준수** 확인
- 기관/소매 거래 대부분이 브로커리지를 통해 이뤄짐

수수료 구조 및 주문 처리 방식

- 마켓 메이커에 주문을 **라우팅**하여 리베이트 수취
- 고객 주문을 내부적으로 상쇄(넷팅)하여 체결 가능
- 모든 주문은 **NBBO(최고 매수/매도 가격)** 내에서 체결돼야 함

브로커별 차이점

- 지원 자산, 주문 유형, 주문 수정 가능 여부 등 차이 존재
- 일부 주문 유형은 **브로커가 미지원**할 수 있음

QuantConnect에서의 주문 처리

- 총 12개 주문 유형 지원

```
# 거래소 개장 시 개장 경매에서 체결을 시도
self.market_on_open_order(symbol, quantity, tag,order_properties)

# stop_price에 도달하면, 시장가 주문을 제출
self.stop_market_order(symbol, quantity, stop_price, tag, order_properties)
```

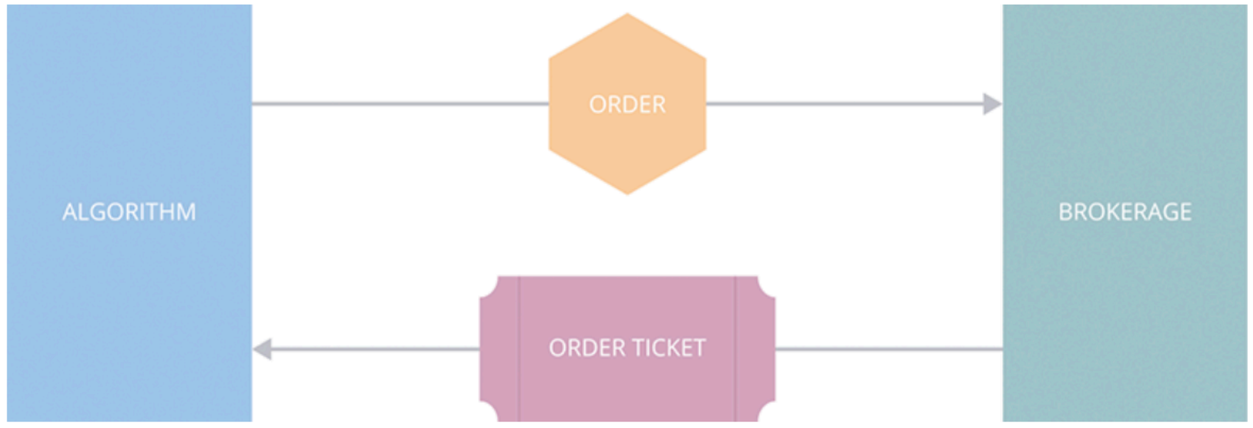


Figure 1.5 Order messaging between the algorithm and brokerage.

자본 시장의 기초 - 브로커리지와 비용

1. 거래 수수료

- 주문당 혹은 주식당 요금 부과
- QuantConnect에서 직접 수수료 모델 설정하여 시뮬레이션 가능

2. 매수-매도 스프레드

- 시장가 주문은 매도/매수 가격을 '건너서' 체결 → 비용 발생
- 실시간 거래 수익성에 중요한 요소

3. 슬리피지(Slippage)

- 기대 체결가와 실제 체결가의 차이
- 주요 원인:
  - 체결 중 가격 변동
- 대량 주문 → "북 워킹" 발생 → 불리한 가격 체결
- QuantConnect에서는 다양한 슬리피지 모델 제공

QuantConnect는 기본적으로 자산이 유동적이라고 가정하지만, 더 미묘한 체결 동작을 사용자 정의하기 위한 여러 대안 모델을 제공합니다:

```
# 주문장 상단에서 즉시 체결 가정
security.set_slippage_model(NullSlippageModel())

# 체결 시 모델-추정 시장 영향
security.set_slippage_model(MarketImpactSlippageModel(self))
```

```
# 이 증권에 대해 $1의 커스텀 수수료 모델 설정
security.set_fee_model(ConstantFeeModel(1))
# 브로커의 모든 수수료 및 제한 설정
self.set_brokerage_model(BrokerageName.INTERACTIVE_BROKERS_BROKERAGE)
```

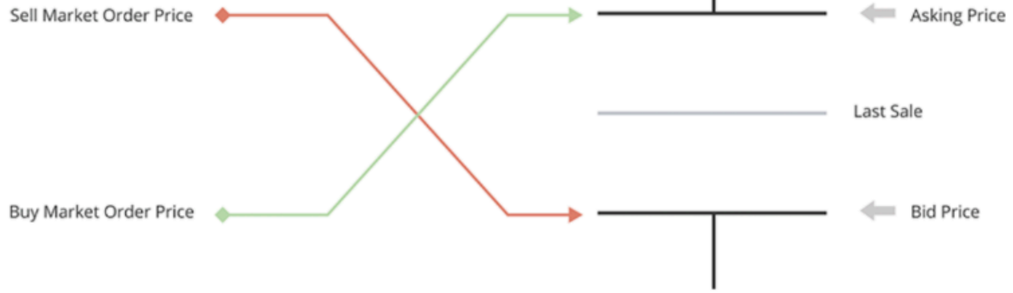


Figure 1.6 Spread crossing: sell market orders fill at the bid price, and buy market orders fill at the ask price. The last sale can be at the bid, at the ask, or somewhere in between them.

Figure 1.7 Slippage costs are different between liquid and illiquid assets.

자본 시장의 기초 - 증권식별자

왜 필요한가?

- 기업 이벤트(이름 변경, 합병, 상장폐지 등)로 인해 **자산 추적 오류 발생 가능**
- 수백 종의 주식을 다루는 퀀트 전략에서는 **일관된 자산 추적**이 필수

주요 전통적 식별자

식별자	설명
CUSIP	미국/캐나다 자산 대상, 8~9자리 코드, 라이센스 필요
FIGI	블룸버그 제공, 무료 사용 가능
ISIN	국제식 표준, CUSIP + 국가코드 조합

- **제약사항**:
  - 비용과 라이선스 문제
  - 커버리지와 정보 완전성 부족
  - 병렬 DB 필요

자본 시장의 기초 - 증권식별자

QuantConnect의 대안: Symbol 시스템

- 자산 정보의 **해시 기반 식별자**
- 데이터베이스 조회 불필요, **자급자족적**
- 255개 국가 & 99개 자산 클래스 지원
- 이름 변경, IPO 등에도 **일관된 식별 유지**

symbol.id # base64 인코딩된 고유 식별자

- 예: SPY R735QTJ8XC9X처럼 표현됨
- 기초 자산 같아도 거래소가 다르면 **다른 심볼로 간주됨** (예: BTCUSD@Coinbase vs BTCUSD@Kraken)

자산과 파생상품 요약

자산 클래스 개요

QuantConnect에서는 5가지 주요 자산 클래스를 지원하며, 대부분 **Algoseek**에서 고품질 데이터를 공급받음.
자산은 **증권(security)** 객체로 관리되고, **시장시간, 마진, 계약승수 등 공통 속성**을 가짐.

1. 미국 주식 (Equity)

- 세계 최대 유동성 시장, 약 **9,000개 상장 기업**
- 기업 이벤트: IPO, 배당, 분할, 이름 변경, 합병 등
- **기초 재무 지표 기반 필터링 가능**
  예: PE 비율, 수익 성장률, FCF 비율, 배당 성향

- PE 비율-자산의 종가를 주당 순이익으로 나눈 값.
- 수익 성장—손익계산서에서 회사 수익의 성장률(%).
- 자유 현금 흐름 비율—영업 현금 흐름의 비율로서의 자유 현금 흐름.
- 배당 지급 비율-순이익 대비 배당금 지급 비율.

QuantConnect에서는 개별 회사의 기본 데이터를 쉽게 가져오고 기본 데이터를 기반으로 주식 스크린을 개발할 수 있습니다.

```
self.add_universe(self._filter) # 필터링된 주식 데이터 유니버스 추가
def _filter(self, fundamental): # pe_ratio를 기반으로 심볼 선택
    return [c.symbol for c in fundamental if c.valuation_ratios.pe_ratio < 10]
```

자산과 파생상품 요약

- 미국 주식 기업 이벤트

분할-회사는 가격이 극적으로 변할 때 시장 참여자들이 더 쉽게 접근할 수 있도록 주식을 나누거나 통합할 수 있습

QuantConnect에서 분할은 거래를 위한 알고리즘 응답을 트리거하거나 연구를 위해 데이터 시리즈에 통합할 수 있는 분할 이벤트 핸들러에 전달되는 이벤트입니다:

```
def on_splits(self, splits): # 전용 분할 이벤트 핸들러
    split = splits.get(self._symbol)
def on_data(self, slice: # on_data 이벤트의 분할
    split = slice.splits.get(self._symbol)
```

배당금 : 회사가 수익성을 통과함에 따라 일부는 주주들에게 배당금을 발행하기로 선택

```
def on_data(self, slice):
    dividend = slice.dividends.get(self._symbol)

def on_dividends(self, dividends):
    dividend = dividends.get(self._symbol)

# 지난 해에 지급된 심볼의 모든 배당금 가져오기
history = self.history[Dividend](symbols,timedelta(365))
```

데이터 정규화 : 가격 분할과 배당금 지급을 고려하기 위해 역사적 가격을 조정 -> 가치 성장을 가능한 한 정확하게 반영하기 위해 역사적 가격을 변경하는 공식을 사용

QuantConnect는 배당금을 포트폴리오에 현금으로 적용하고 언제 그리고 어떻게 재투자할지 결정할 수 있게 하는 원시 모드를 지원합니다.
QuantConnect에서는 자산을 전략에 추가하는 동안 데이터 정규화 모드를 선택합니다:

```
# 역사적 가격의 조정 없음, 배당금은 현금으로 지급
self.add_equity("AAPL",data_normalization_mode=DataNormalizationMode.RAW)
# 역사적 가격의 전체 분할 및 배당금 조정
self.add_equity("AAPL", data_normalization_mode=DataNormalizationMode.ADJUSTED)
# 분할에 대해서만 조정하고, 배당금을 현금으로 지급
self.add_equity("AAPL", data_normalization_mode=DataNormalizationMode.SPLIT_ADJUSTED)
```

자산과 파생상품 요약

- 증권 변경-IPO, 이름 변경, 분사, 합병 및 상장 폐지 - 회사의 정규 활동은 퀀트에게 많은 모델링 과제를 생성함

IPO-회사가 준비되면, 거래소에 공개적으로 상장을 신청할 수 있습니다. 승인되면 IPO로 공식적으로 상장됩니다. 자산이 알고리즘에 추가될 때, 새로운 상장의 세부 정보와 함께 증권 변경 이벤트가 생성됩니다.

```
def on_securities_changed(self, changes):
    for security in changes.added_securities:
        Pass
```

이름 변경-회사는 종종 새로운 사업 라인을 반영하거나 더 나은 시장 티커로 교체하기 위해 리브랜딩합니다.

이러한 티커 변경은 심볼 변경 이벤트에서 처리되며, 전략에 새로운 티커와 이전 티커를 제공합니다.

QuantConnect는 자산을 추적하기 위해 심볼 객체를 사용하므로 새 티커로 전환할 필요가 없습니다.

```
def on_symbol_changed_events(self, changes):
    for symbol, change in changes.items():
        self.log(f"Change: { change.old_symbol} ->{change.new_symbol}”)
```

- 분사 및 합병-회사는 때때로 작은 회사를 인수하여 거래에서 작은 회사를 흡수합니다. 이는 작은 회사의 단순한 상장 폐지로 모델링됩니다. 합병은 비슷하지만 새로운 티커를 초래할 수 있습니다; 이는 상장 폐지와 이름 변경으로 모델링됩니다.
- 상장 폐지-대부분의 자산 클래스에는 일종의 상장 폐지가 있습니다. 주식의 경우, 이는 공개 시장에서 회사가 제거되는 것입니다. 이벤트는 상장 폐지 이벤트 핸들러에서 캡처됩니다. 또한, 제거된 증권으로 증권 변경 이벤트에 표시됩니다.

```
def on_delistings(self, delistings): # 자산 상장 폐지 이벤트
    delisting = delistings.get(self._symbol)

def on_securities_changed(self, changes): # 선택된 스크린에서 제거
    for security in changes.removed_securities:
        self.log(f'Security {security.Symbol} removed from universe')
```

자산과 파생상품 요약

2. 주식 옵션 (Equity Options)

- 의미: 특정 미래 날짜(만기일)에 정해진 가격(행사가)으로 기초 자산(주식)을 살(콜) 또는 팔(풋) 권리를 가진 계약 (의무는 없음).
- 주요 속성

항목	설명
기초 자산	옵션이 기반한 주식 (예: AAPL)
옵션 권리	콜: 매수 권리 / 풋: 매도 권리
행사가 (Strike)	계약 가격 – 옵션의 가치 결정 요소
만기일 (Expiry)	행사 가능 마지막 날 (미국식: 언제든 행사 가능)
계약 승수	대부분 1계약 = 100주

- 내가격 vs 외가격
  - **ITM (In The Money)**:
    - 콜: 주식 가격 > 행사가
    - 풋: 주식 가격 < 행사가 → 행사 시 이익
  - **OTM (Out of The Money)**:
    - 수익 없음, 보험처럼 사용됨

각 기초 자산에는 수백 개의 옵션 계약이 있으며, 각 자산을 기반으로 유니버스를 형성합니다.

```
option = self.add_option("SPY") # SPY에 대한 옵션 유니버스 요청
option.set_filter( ... ) # 반환된 옵션 계약 필터링
self.add_option_contract(contract_symbol) # 특정 옵션 계약
def on_data(self, slice): # on_data에서 옵션 체인 사용
    chain = slice.option_chains.get(option.symbol)
```

자산과 파생상품 요약

3. 지수 옵션 (Index Options)

- **QuantConnect 는 주간 버전과 함께 세 가지 지수 옵션 월간 옵션 체인을 지원 : SPX, VIX, NDX** 등 지수 대상
- **유럽식 옵션**: 만기일에만 행사, **현금 결제, 지급액은 지수 수준과 행사 수준의 차이에 계약 승수를 곱한 값으로 계산함**

이러한 지수는 다음과 같이 알고리즘에 추가될 수 있습니다:

```
self._index_symbol = self.add_index('SPX').symbol # 알고리즘에 지수 추가
option = self.add_index_option(self._index_symbol) # 지수에 옵션 추가
option.set_filter(-2, 2, 0, 90) # 계약 필터링
```

자산과 파생상품 요약

4. 미국 선물 (Futures)

- 상품/지수/채권/통화 등 다양한 파생상품
- **계약 롤오버와 연속 가격 시리즈** 중요
- QuantConnect는 **160개 이상 지원**
- 실시간 거래 시, **mapped 자산**을 사용해야 함
- 설정 예시:

각 상품에는 개별 만료일을 나타내는 선물 계약 모음이 있으며, 유니버스를 형성합니다:

```
future = self.add_future(Futures.Indices.SP_500_E_MINI) # ES 선물 추가
future.set_filter(0, 90) # 90일 필터링
```

QuantConnect에서, 이 연속 최근 계약은 표준 계약으로 참조됩니다. 현재 기초, 거래 가능한 계약은 맵핑된 자산이라고 불립니다.

이는 다음과 같이 액세스할 수 있습니다:

```
future = self.add_future(Futures.Indices.SP_500_E_MINI) # 표준 자산
adjusted_price = self.security [future.symbol].price # 조정된 가격
raw_price = self.security[future.mapped].price # 원시 가격
```

거래를 라우팅할 때, 연속 계약은 거래할 수 없으므로 기초 맵핑된 자산을 사용해야 합니다.

```
self.market_order(future.mapped, 1) # 맵핑된 자산의 1 계약 구매
```

5. 암호화폐 (Crypto)

- 빠르게 성장한 **글로벌 분산 자산 클래스**
- **중앙화 vs DEX 거래소** 구분
- **거래소별로 별개 시장 취급**
- API 예시:

서로 다른 가격을 가지고 있고 중앙화된 피드가 없기 때문에, 암호화폐 거래소는 다른 시장으로 간주됩니다. 이들은 add crypto API로 추가할 수 있습니다:

```
# 시장별로 두 거래소의 가격 피드 요청
coinbase_btcusd = self.add_crypto("BTCUSD",market=Market.COINBASE).symbol
kraken_btcusd = self.add_crypto("BTCUSD",market=Market.KRAKEN).symbol
```

각 거래소는 지원되는 마진, 자산, 주문 유형이 다르므로, 관련 거래소에 대한 브로커리지 모델을 설정하는 것이 중요합니다. 여기서 계정 유형(현금 또는 마진)도 설정할 수 있습니다.

```
self.set_brokerage_model(BrokerageName.KRAKEN, AccountType.MARGIN)
```