



QuantConnect – A Complete Guide

· 38 min read



21951 total views

Last Updated on May 14, 2021

Get 10-day Free Algo Trading Course

Table of contents

1. [What is QuantConnect?](#)
2. [Why should I use QuantConnect?](#)
 - [Backtesting and Live trading](#)
 - [Paper trading](#)
 - [Free data](#)
 - [Alpha Stream](#)
 - [Strategy Development Framework](#)
 - [Forum](#)
3. [Why Shouldn't I use QuantConnect?](#)
4. [Is QuantConnect free?](#)
5. [Does QuantConnect support Python?](#)
6. [How do I get started with QuantConnect?](#)
 - [Signing-up and membership](#)
 - [Docs](#)
 - [Bootcamp](#)
7. [The 2 ways to Backtest Strategies on QuantConnect – Classic vs SDF](#)
8. [The Lab/Terminal and Creating your First Algorithm](#)
 - [Creating your First Algorithm](#)
 - [Basic Structure](#)
 - [Initialize and OnData Methods](#)
 - [Initialize\(\)](#)
 - [Setting Dates and Cash](#)
 - [Adding Data](#)

Join over 10,000 future traders and get our **10-day "Know-What-To-Google" Algo Trading email course**. 1 short email a day for 10 days.

"I have not read it, but I'm sure it is great." - My girlfriend

Your first name

Your email address

Give me the 10-day crash course!

- [How can I update orders using QuantConnect?](#)
- [How can I get historic data using QuantConnect?](#)
- [How can I get fundamental data using QuantConnect?](#)
- 10. [How to Create and Backtest a Strategy in QuantConnect?](#)
 - [Our First Strategy! Mean Reversion on Lean Hog Futures \(Classic Backtesting\)](#)
- 11. [What is QuantConnect's Strategy Development Framework \(SDF\)?](#)
 1. [Step 1: Using QuantConnect's Universe Selection example: Tech stocks](#)
 2. [Step 2: Using QuantConnect's Alpha Creation example: Smart Insider](#)
 - [Overview](#)
 - [Generating Insights](#)
 - [Implementation](#)
 3. [Step 3: Using QuantConnect's Portfolio Construction example: Insight Weighted](#)
 4. [Step 4: Using QuantConnect's Execution Engine example: Immediate](#)
 5. [Step 5: Using QuantConnect's Risk Management example: Portfolio Drawdown](#)
 6. [Our Second Strategy! Backtesting Tech Stocks using Insider Insights \(SDF Backtesting\)](#)
- 12. [Final Thoughts](#)
- 13. [Link to download code](#)

What is QuantConnect?

QuantConnect is an algorithmic trading browser-based platform that lets you develop, test and execute strategies.

Join over 10,000 future traders and get our **10-day "Know-What-To-Google" Algo Trading email course**. 1 short email a day for 10 days.

"I have not read it, but I'm sure it is great." - My girlfriend

Your first name

Your email address

Give me the 10-day crash course!

QuantConnect such as their ~~Strategy Development Framework~~ (a series of pre-made plug and play modules covering key aspects of an algorithmic strategy that can be utilised in many different combinations) and their **Alpha Stream**, which is a feature that lets you attempt to monetise any strategies you might have by “leasing” them out to interested 3rd parties- who can see the insights and trading recommendations generated by your algorithms and also utilise them, without actually seeing their underlying code.

Why should I use QuantConnect?

- Backtesting and Live trading
- Paper trading
- Free data
- Strategy development framework
- Alpha stream
- Forum

Backtesting and Live trading

Most platforms and APIs allow you to easily execute a live trading strategy through them, however a unique aspect of QuantConnect is just how easy it is to give your strategies a thorough back-test using the platform.

All you have to do is configure a start date and end date for the back test period, an initial cash amount to allocate to the strategy and hit “Backtest” – and you’re off!

Join over 10,000 future traders and get our **10-day "Know-What-To-Google" Algo Trading email course**. 1 short email a day for 10 days.

"I have not read it, but I'm sure it is great." - My girlfriend

Your first name

Your email address

Give me the 10-day crash course!

Paper trading

When deploying a strategy to live, it is possible to elect to “paper trade” (trade using pretend money) to test your strategies before committing to using real money.

This is fantastically useful both if you are a novice to trading in general, and for all traders- beginners and experienced alike- to get a better feel that your trading strategy actually is profitable on out of sample data before committing to it.

Paper trading can be chosen in the “Select Brokerage” step of the *Go Live* flow.

Free data

QuantConnect offers a huge amount of free data through the [QuantConnect Data Explorer](#).

Note that whilst this is all free to use within the IDE, it is not necessarily all free to download and use in an external environment- sometimes you will have to pay for that.

We will show you later on how to use any of this data in your algorithms/within QuantConnect’s Jupyter Notebooks for data exploration.

Join over 10,000 future traders and get our **10-day "Know-What-To-Google" Algo Trading email course**. 1 short email a day for 10 days.

"I have not read it, but I'm sure it is great." - My girlfriend

Your first name

Your email address

Give me the 10-day crash course!

this prediction, actively ignoring ~~position sizing, portfolio management~~ and risk.

Any interested party/fund is able to rent usage of the Alpha either at a “Shared Price” (where others are able to use the same Alpha also) or an “Exclusive Price” which locks out anybody else from using the same Alpha.

They are then able to use the Alpha to trade their own funds or test it on out of sample data, but are prevented from seeing the source code. QuantConnect takes special care to make Alpha's to be as hard to reverse-engineer as possible.

The marketplace shows a bunch of key stats to compare strategies by, helping purchases make an informed decision.

You can read a bit more about creating Alphas [here](#), but we will show you how to create some later anyway in the [Using QuantConnect's Alpha Creation example: Smart Insider / Generating Insights](#) sections of the guide.

Strategy Development Framework

QuantConnect's [Strategy Development Framework](#) (SDF) comprises of plug and play modules designed to make replicating, sharing and re-using specific components of strategies as easy as possible.

It comprises of 5 sections–

Join over 10,000 future traders and get our **10-day "Know-What-To-Google" Algo Trading email course**. 1 short email a day for 10 days.

"I have not read it, but I'm sure it is great." - My girlfriend

Your first name

Your email address

Give me the 10-day crash course!

price to move X standard deviations above or below the recent mean before executing a trade).

- **Risk Management** (modules focused on minimising risks and cutting losing positions early).

As you might have noticed, there is a clear directional flow between the modules.

This may all seem a bit overwhelming at first, but don't worry- we'll cover everything in depth with examples in the [What is QuantConnect's strategy development framework?](#) section of the guide!

Forum

Finally, QuantConnect has its very own [forum](#).

You can discuss and share strategies and tests with fellow traders, get help with various known data issues, or check out or participate in the various competitions going on.

Why Shouldn't I use QuantConnect?

- Learning Curve
- Not completely free
- Your work is stored on their servers
- No Optimizer

Join over 10,000 future traders and get our **10-day "Know-What-To-Google" Algo Trading email course**. 1 short email a day for 10 days.

"I have not read it, but I'm sure it is great." - My girlfriend

Your first name

Your email address

Give me the 10-day crash course!

However if you just want to dive straight into testing an algorithm, you might prefer to go with a simpler setup you are already familiar with.

Not completely free

Also, not every component of QuantConnect is entirely free- we'll cover exactly what is and isn't in just a moment!

Your work is stored on their servers

Your code is stored on QuantConnect's servers as opposed to your local machine.

Thus, you might be wary of security risks and your code being accessed without your permission.

That said, QuantConnect seems to take security seriously. See their statement [here](#).

On a related note, QuantConnect allows you to self-host their fully open-source algorithmic trading engine, [LEAN](#).

No optimizer

Finally, QuantConnect does not have an optimizer allowing for mass back-testing of different parameters/parameters in a range to find the optimal, nor any sort of walk-forward optimizer.

Join over 10,000 future traders and get our **10-day "Know-What-To-Google" Algo Trading email course**. 1 short email a day for 10 days.

"I have not read it, but I'm sure it is great." - My girlfriend

Your first name

Your email address

Give me the 10-day crash course!

100%

The minimum investment to be able to live trade on QuantConnect is \$8/month for the Quant Researcher membership and a further \$20/month for an L-Micro live node (1 CPU/0.5GB RAM).

You can further upgrade your back testing node and live trading node and also purchase a research node of various tiers at additional expense to speed up your back testing/allow parallel back testing.

It can be worth spending at least \$40/month in total on various components as that unlocks QuantConnect's "Bronze tier" which will give you access to email support with 4 available tickets per month.

Go [here](#) to see the plans.

QuantConnect also offer a massive amount of historical data which is free to use within the IDE, but you may have to pay on a per dataset basis to download outside of QuantConnect.

Finally if you are a student, QuantConnect very kindly offers a free year's access to their researcher tier!

Join over 10,000 future traders and get our **10-day "Know-What-To-Google" Algo Trading email course**. 1 short email a day for 10 days.

"I have not read it, but I'm sure it is great." - My girlfriend

Your first name

Your email address

Give me the 10-day crash course!

Signing up and membership

First, head on over to the [sign-up](#) page and create an account.

If you just want back test you need do nothing more. Head over to the [lab/terminal](#) and let's get started building some algorithms!

If you want to live trade head over to the [account upgrade](#) page and purchase yourself a Quant Researcher membership and a live trading node- though you can wait till after back testing some strategies to do this if you want.

Docs

We are going to try and give you a crash course in some of the key functionality of QuantConnect in this guide, but in case you need further help and/or want to learn more in general, QuantConnect has some very thorough [documentation](#).

Bootcamp

Equally useful is the [Bootcamp](#), which you can find in the lab/terminal section of QuantConnect.

Here you will be guided step by step through some code/algorithms covering basic functionality on QuantConnect, with hints and full solutions provided.

The lessons can be very useful to refer to if you forget exactly how to implement something, or just want a template to adapt slightly for your specific use case.


Join over 10,000 future traders and get our **10-day "Know-What-To-Google" Algo Trading email course**. 1 short email a day for 10 days.

"I have not read it, but I'm sure it is great." - My girlfriend

Your first name

Your email address

Give me the 10-day crash course!

- 
1. **Universe Selection** (pick assets to trade)
 2. **Alpha Creation** (create trading signals)
 3. **Portfolio Construction** (set the targets for each asset to hold)
 4. **Execution** (execution method to reach targets)
 5. **Risk Management** (set logic to liquidate elements of portfolio during poor performance)

We will cover SDF in detail in a later section.

The Lab/Terminal and Creating your First Algorithm

Creating your First Algorithm

The Lab/Terminal is where you will write all your algorithms.

There used to be a distinct divide between “classic” (only your own code- no SDF modules) and “framework”/SDF algorithms on QuantConnect, however nowadays the separation is more fluid.

Click on “Create new Algorithm” on the left hand panel to get started.

As you can see in the right hand side of the image below, **main.py** is where all your code goes.

Join over 10,000 future traders and get our **10-day "Know-What-To-Google" Algo Trading email course**. 1 short email a day for 10 days.

"I have not read it, but I'm sure it is great." - My girlfriend

Your first name

Your email address

Give me the 10-day crash course!



The basic process is to select the SDF modules you want to use (or skip some/all of them if you prefer), then press “Create Algorithm” in the bottom right.

You can add as many Alpha modules as you want, but you can only select a single module for the other sections of the framework.

Once you’ve pressed *Create Algorithm*, you’ll be able to write code manually within **main.py**.

Basic Structure

If you create an algorithm with some SDF modules you should be left with a **main.py** that looks something like this:

If you were to not use any SDF modules, you’d be left with the most basic setup possible that looks like this:

Let’s go with the basic setup for now to explain things.

The first line `class CalibratedUncoupledCoreWave(QCAlgorithm):` creates our algorithm.

Join over 10,000 future traders and get our **10-day "Know-What-To-Google" Algo Trading email course**. 1 short email a day for 10 days.

"I have not read it, but I'm sure it is great." - My girlfriend

Your first name

Your email address

Give me the 10-day crash course!



You can of course write your own additional methods/functions and get them to activate at certain times in `Initialize()` or `OnData()` – we'll explain as we go along!

Initialize()

`def Initialize(self):` is run before back testing/live trading commences. In it we set important variables, modules, add data and warm up indicators and so forth.

We can also use **Scheduled Events** in `Initialize()` to trigger code to run at specific times of the day.

Setting Dates and Cash

Here we can see it has `self.SetStartDate(2018, 4, 1)` and `self.SetCash(100000)` methods. These set the starting date for a back test and the cash allocated to the algorithm to begin with.

You can also add `self.SetEndDate()` to determine the end date of a back test. If this is not set, the back test runs until the present date by default.

Join over 10,000 future traders and get our **10-day "Know-What-To-Google" Algo Trading email course**. 1 short email a day for 10 days.

"I have not read it, but I'm sure it is great." - My girlfriend

Your first name

Your email address

Give me the 10-day crash course!



We also add data within the `Initialize()` method.

The general template to do so is very simple.

For equities we use `self.AddEquity()` which returns a security object that we can assign internally for future use, for example with the SPY:

Python

```
1 | self.spy = self.AddEquity("SPY", Resolution.Hour, Market.Oanda)
```

Resolution is the time period of a data bar, and the default options for Equities are:

- `Resolution.Tick`
- `Resolution.Second`
- `Resolution.Minute`
- `Resolution.Hour`
- `Resolution.Daily`

The **Market** parameter dictates which market the data is drawn from, in case multiple markets track the same asset and you wish to take data from a specific one. QuantConnect has default markets it uses for each asset if you do not specify anything.

There are also:

Join over 10,000 future traders and get our **10-day "Know-What-To-Google" Algo Trading email course**. 1 short email a day for 10 days.

"I have not read it, but I'm sure it is great." - My girlfriend

Your first name

Your email address

Give me the 10-day crash course!

method we use throughout this guide.

If you ever wish to get a full run down of available functionality, click on "API" on the right hand panel of the terminal and search the relevant method/term, as shown below:

Setting Indicators

We would also set up indicators that we wanted to use in the main algorithm in `Initialize()`, such as RSI or MACD.

For example for an RSI indicator, we would set key variables such as the look back period and overbought and oversold levels, remembering to set them to `self` so that they are referenceable throughout different methods in the algorithm.

```
1 | RSI_Period      = 14          # RSI Look back period
2 | self.RSI_OB    = 60          # RSI Overbought level
3 | self.RSI_OS    = 40          # RSI Oversold level
```

Python

We would then set up an RSI indicator object (which is inherited from the `QCAAlgorithm` class) for the SPY using our desired RSI look back period:

```
1 | self.RSI_Ind_SPY = self.RSI("SPY", RSI_Period)
```

Python

Join over 10,000 future traders and get our **10-day "Know-What-To-Google" Algo Trading email course**. 1 short email a day for 10 days.

"I have not read it, but I'm sure it is great." - My girlfriend

Your first name

Your email address

Give me the 10-day crash course!

```

2 self.SetWarmUp(1000000, // warm up 1 day of data
3
4 # Or perhaps ensure that the RSI indicator has enough data before
5 trading.
  self.SetWarmUp(RSI_Period)

```

Also note, algorithms can use the boolean `IsWarmingUp` to determine if the warm-up period has completed.

Scheduled Events

You can schedule codeblocks (methods/functions) to run at specific times of a day independent of when your algorithm receives a data update with **Scheduled Events**.

Scheduled events require a date and time rule to specify when the event is fired alongside a function/method to fire, and go inside `Initialize()`.

The method to schedule an event is `self.Schedule.On(DateRule, TimeRule, Action)`.

Here are some example **DateRules** and **TimeRules**.

To see full lists go to [scheduled events](#) documentation and expand the DateRule and TimeRule documentation:

Join over 10,000 future traders and get our **10-day "Know-What-To-Google" Algo Trading email course**. 1 short email a day for 10 days.

"I have not read it, but I'm sure it is great." - My girlfriend

Your first name

Your email address

Give me the 10-day crash course!

In summary, this is what our algorithm structure might look like now having filled out `Initialize()` with a bunch of the features we've just learnt:

OnData()

`def OnData(self, data):` is activated each time new data is passed to your algorithm, so this can be hourly, daily or weekly etc. depending on the data resolution you request in `Initialization(self)`.

You might fire trading logic in here, or update indicators or dataframes.

Remember you can also activate functions on regular time intervals/certain events not related to a data update with **Scheduled Events**.

There are also other functions like `OnHour()` or `OnEndOfDay()` inbuilt to QuantConnect that fire on their respective timeframes. We will gradually introduce these as the guide goes on to keep things manageable.

Back testing vs Live trading

Writing code for back testing and live trading is exactly the same- it all goes in the `main.py` panel.

Join over 10,000 future traders and get our **10-day "Know-What-To-Google" Algo Trading email course**. 1 short email a day for 10 days.

"I have not read it, but I'm sure it is great." - My girlfriend

Your first name

Your email address

Give me the 10-day crash course!



Also note there is a useful `self.LiveMode()` boolean that your algorithm can use to tell if it's live. For instance:

```
1 | if self.LiveMode:
2 |     # execute this code only if algorithm is in live trading mode
```

Python

Research (Jupyter notebooks)

Finally, QuantConnect comes with internal Jupyter notebook environments that you can access within your projects.

These can be useful for data exploration- we will use them later on in this guide to demonstrate how to get and visualize historical and fundamental data.

What are the basic Backtesting operations in QuantConnect?

These methods are mainly meant for the Classic backtesting method.

How can I place orders using QuantConnect?

QuantConnect allows you to place a variety of orders.

Join over 10,000 future traders and get our **10-day "Know-What-To-Google" Algo Trading email course**. 1 short email a day for 10 days.

"I have not read it, but I'm sure it is great." - My girlfriend

Your first name

Your email address

Give me the 10-day crash course!

```
marketOrderTicket = self.marketOrder( 500 , 100 )
```

Market orders execute immediately and buy up or down the order book starting from the current market price until filled, so be aware of potential slippage on larger orders in less liquid markets.

Since market orders theoretically execute almost immediately, you're not likely to want or be able to cancel or update them.

You could however use the marketOrderTicket object to check the average fill price of your market object like so:

Python

```
1 | # Check the average fill price of your market order using your
2 | OrderTicket
   | self.Debug("Market Order Fill Price: {0}".format(marketOrderTicket
   | .AverageFillPrice))
```

Note that in the QuantConnect terminal any code you have will **pause for 5 seconds** after executing a market order, to give time for the order to fill.

If you wish to adjust how long this pause is, use the following line of code—changing the seconds value as desired:

Join over 10,000 future traders and get our **10-day "Know-What-To-Google" Algo Trading email course**. 1 short email a day for 10 days.

"I have not read it, but I'm sure it is great." - My girlfriend

Your first name

Your email address

Give me the 10-day crash course!

```
2 | self.MarketOrder("SPY", 100, True)
```

To *sell* instead of buy we must use a **negative quantity**, like so:

Python

```
1 | # Create a Market Order to sell 100 shares of SPY asynchronously
2 | self.MarketOrder("SPY", -100, True)
```

Limit orders do not fill immediately- rather only when the specified buy or sell price is reached

Here is an example of how to place a limit order that executes 5% below the current price (5% below the close price of the last data bar):

Python

```
1 | # Purchase 10 SPY shares when its 5% below the current price
2 | close = self.Securities["SPY"].Close
3 | limitTicket = self.LimitOrder("SPY", 10, close * .95)
```

How can I set stop losses and take profits using QuantConnect?

Stop Losses

Unfortunately QuantConnect does not allow the usage of trailing stop losses.

Join over 10,000 future traders and get our **10-day "Know-What-To-Google" Algo Trading email course**. 1 short email a day for 10 days.

"I have not read it, but I'm sure it is great." - My girlfriend

Your first name

Your email address

Give me the 10-day crash course!



Python

```
1 | stopMarketTicket = self.StopMarketOrder("IBM", -10, 18)
```

And this is an example of how we could set a limit order to buy 10 shares of SPY 5% below the current price, and set a market stop loss for all 10 shares a further 3% below our limit buy to protect ourselves from downside:

Python

```
1 | # Purchase 10 SPY shares when its 5% below the current price
2 | current_price= self.Securities["SPY"].Close
3 | limitTicket = self.LimitOrder("SPY", 10, current_price* .95)
4 |
5 | # And set stop loss 3% below purchase price
6 | stopMarketTicket = self.StopMarketOrder("SPY", -10, current_price*
   | 0.95*0.97)
```

Take Profits

To take profits once an asset has risen in value a pre-determined amount from our buy in points, we want to set *limit sell orders*.

This is how we could set a take profit for 10shares of SPY at \$350(again remember the *negative quantity*):

Join over 10,000 future traders and get our **10-day "Know-What-To-Google" Algo Trading email course**. 1 short email a day for 10 days.

"I have not read it, but I'm sure it is great." - My girlfriend

Your first name

Your email address

Give me the 10-day crash course!

```
3 | limitTicket = self.LimitOrder("SPY", 10, current_price* .95)
4 |
5 | # Set stop loss 3% below purchase price
6 | stopMarketTicket = self.StopMarketOrder("SPY", -10, current_price*
7 | 0.95*0.97)
8 |
9 | # And set take profits 5% and 10% above purchase price
10 | tp1Ticket = self.LimitOrder("SPY", -5, current_price*1.05)
    | tp2Ticket = self.LimitOrder("SPY", -5, current_price*1.1)
```

Note that in practice you'd probably want to update or delete the stop loss if either of the take profits hit, or delete the take profits if the stop loss hits- we'll show you how to in the next section!

How can I cancel orders or Liquidate my Portfolio using QuantConnect?

Cancelling orders

We can cancel an order simply by using the

`Cancel(optionalDescriptiveTextString)` method of the OrderTicket object.

This is how we can set a limit buy order, and later on decide to cancel it:

Join over 10,000 future traders and get our **10-day "Know-What-To-Google" Algo Trading email course**. 1 short email a day for 10 days.

"I have not read it, but I'm sure it is great." - My girlfriend

Your first name

Your email address

Give me the 10-day crash course!

Python

```
1 # Use order response object to read status
2 if response.IsSuccessful:
3     self.Debug("Order successfully cancelled")
```

Also, you can cancel all open orders using

`Self.Transactions.CancelOpenOrders()` :

Python

```
1 # Cancel all open orders
2 allCancelledOrders = self.Transactions.CancelOpenOrders()
```

Or you can cancel all open orders only related to a particular asset by providing its string to `CancelOpenOrders()` , like so:

Python

```
1 # Cancel orders related to SPY
2 spyCancelledOrders = self.Transactions.CancelOpenOrders("SPY")
```

So for instance to extend our previous example, if our stop loss hit we could cancel all remaining open orders for SPY (which would be the take profit limit orders at that point) to clean things up- allowing us to simulate a **OCA/One-Cancels-All** style of order in a round about way (QuantConnect doesn't allow direct OCA orders).

Liquidation

Join over 10,000 future traders and get our **10-day "Know-What-To-Google" Algo Trading email course**. 1 short email a day for 10 days.

"I have not read it, but I'm sure it is great." - My girlfriend

Your first name

Your email address

Give me the 10-day crash course!

THIS BOTH **cancels all open orders** for the asset/whole portfolio then creates **market sell orders** for your holdings of the asset/whole portfolio- returning you 100% to cash in record speed.

How can I update orders using QuantConnect?

To update an order you must use its OrderTicket.

You can update the following attributes of an order:

Orders are updated by passing a **UpdateOrderFields** object to the **Update()** method of the OrderTicket.

Lets create an order we will then update:

Python

```
1 | # Create an order
2 | limitTicket = self.LimitOrder("SPY", 10, 221)
```

To update the order first create an UpdateOrdersField object:

Python

```
1 | # Create an UpdateOrderFields object
2 | updateSettings = UpdateOrderFields()
```

Join over 10,000 future traders and get our **10-day "Know-What-To-Google" Algo Trading email course**. 1 short email a day for 10 days.

"I have not read it, but I'm sure it is great." - My girlfriend

Your first name

Your email address

Give me the 10-day crash course!

Python

```
1 response = limitTicket.Update(updateSettings)
2
3 # Validate the response is OK
4 if response.IsSuccessful:
5     self.Debug("Order updated successfully")
```

We've now gone through some of the key functionality of the order placement and management system on QuantConnect, but do check out the [trading-and-orders documentation](#) for a full run down of what is possible!

How can I get historic data using QuantConnect?

Firstly, its important to note you can access historic data either directly in your algorithm in the terminal, or in a Research Notebook.

In both cases you will use the `History(symbol[], time period/bar period/start + end time period, resolution = null)` method and the historic data is returned in a pandas dataframe.

- **Symbol[]** can be a single string or a list of strings
- **time period/bar period/start + end time period** is a little confusing because QuantConnect's API lets you express the time period in three different ways- we will give some examples in a sec!
- **resolution** is the time period of a data bar/row in your dataframe- Minute/Hour/Daily etc.

Join over 10,000 future traders and get our **10-day "Know-What-To-Google" Algo Trading email course**. 1 short email a day for 10 days.

"I have not read it, but I'm sure it is great." - My girlfriend

Your first name

Your email address

Give me the 10-day crash course!

Python

```
1 # Returns the past 10 days of historical hourly data
2 self.df= self.History(self.Symbol("SPY"), timedelta(days=10),
    Resolution.Hour)
```

Python

```
1 # Returns the past 10 bars of historical hourly data
2 self.df= self.History(self.Symbol("SPY"), 10, Resolution.Hour)
```

Python

```
1 start_time = datetime(2019, 1, 1) # start datetime for history call
2 end_time = datetime(2020, 1, 1) # end datetime for history call
3
4 # Returns hourly historical data between January 1st 2019 and
5 January 1st 2020
self.df= self.History(self.Symbol("SPY"), start_time, end_time,
    Resolution.Hour)
```

In a Research Notebook:

Very similar except instead of referring to self, you need to create a **QuantBook()** object in the notebook and then refer to that.

Join over 10,000 future traders and get our **10-day "Know-What-To-Google" Algo Trading email course**. 1 short email a day for 10 days.

"I have not read it, but I'm sure it is great." - My girlfriend

Your first name

Your email address

Give me the 10-day crash course!



Python

```
1 | # Returns the past 10 days of historical hourly data
2 | df = qb.History(spy.Symbol, timedelta(days=10), Resolution.Hour)
```

Now lets pretend we are back within the terminal/an algorithm and show you a few different things we can do with historic data.

Multiple Tickers in a Dataframe

We can access more than one ticker's data in the same dataframe:

Python

```
1 | # Subscribe to data from multiple tickers
2 | self.AddEquity("IBM", Resolution.Daily)
3 | self.AddEquity("AAPL", Resolution.Daily)
4 |
5 | # Set start and end time.
6 | start_time = datetime(2019, 04, 25) # start datetime for history
7 | call
8 | end_time = datetime(2020, 04, 27) # end datetime for history call
9 |
   | self.dataframe = self.History([self.Symbol("IBM"),
   | self.Symbol("AAPL")], start_time, end_time)
```

Note that because we didn't select a *Resolution*, QuantConnect's API used the default for securities which is daily.

Join over 10,000 future traders and get our **10-day "Know-What-To-Google" Algo Trading email course**. 1 short email a day for 10 days.

"I have not read it, but I'm sure it is great." - My girlfriend

Your first name

Your email address

Give me the 10-day crash course!

*Comparing
daily
"close"
prices for
AAPL and
IBM*

How can I get fundamental data using QuantConnect?

Unfortunately QuantConnect doesn't offer direct access to fundamental historical data within algorithms.

That said, you can still filter assets you'd like based on current fundamental data using the Coarse and Fine Universe selection modules from the SDF in an algorithm.

Also, you can access historical fundamental data in Research notebooks- so let's do that!

GetFundamental()

To grab historic fundamental data we are always going to use the `qb.GetFundamental(Symbols, Selector, StartDate, EndDate)` method.

Symbols, StartDate and EndDate behave as in the `History()` method we just used.

Join over 10,000 future traders and get our **10-day "Know-What-To-Google" Algo Trading email course**. 1 short email a day for 10 days.

"I have not read it, but I'm sure it is great." - My girlfriend

Your first name

Your email address

Give me the 10-day crash course!

a few stocks are.

Firstly, lets subscribe to some symbols:

Python

```
1 | qb = QuantBook()  
2 | amzn = qb.AddEquity("AMZN")  
3 | goog = qb.AddEquity("GOOG")  
4 | ibm = qb.AddEquity("IBM")
```

And then add a start and end date and call the `GetFundamental()` method using the `ValuationRatios.PERatio` selector:

Python

```
1 | start_time = datetime(2020, 1, 1) # January 1st 2020  
2 | end_time = datetime.now() # Today's date  
3 |  
4 | # Get the PE ratio for all securities between given dates  
5 | pe_history = qb.GetFundamental(qb.Securities.Keys,  
6 | "ValuationRatios.PERatio", start_time, end_time)  
7 |  
   pe_history
```

This is what that looks like:

We could plot the P/E ratios overtime to make visualisation easier using:

Join over 10,000 future traders and get our **10-day "Know-What-To-Google" Algo Trading email course**. 1 short email a day for 10 days.

"I have not read it, but I'm sure it is great." - My girlfriend

Your first name

Your email address

Give me the 10-day crash course!



Python

```
1 | # Sort stocks by their average PE ratio
2 | sorted_by_mean_pe = pe_history.mean().sort_values()
3 | sorted_by_mean_pe
```

```
IBM R735QTJ8XC9X      12.942418
G00CV VP83T1ZUHR0L    29.339131
AMZN R735QTJ8XC9X     107.074606
dtype: float64
```

This of course would be much more useful with a much bigger list of stocks!

Finally, we can filter down the dataframe to display results of only a single ticker by simply referencing that ticker in square brackets from the dataframe like so:

Python

```
1 | pe_history["AMZN R735QTJ8XC9X"]
```

You can investigate absolutely any other fundamental data using exactly the process as with `ValuationRatios.PERatio` here – just change the selector!

How to Create and Backtest a Strategy in QuantConnect?

Join over 10,000 future traders and get our **10-day "Know-What-To-Google" Algo Trading email course**. 1 short email a day for 10 days.

"I have not read it, but I'm sure it is great." - My girlfriend

Your first name

Your email address

Give me the 10-day crash course!

price rapidly falls significantly below the recent average and sell when it rises significantly above.

The idea is that substantial deviations from a trend line are likely to be only temporary in either direction even if the overall trend continues to hold.

Such a strategy can be somewhat thought of as “arbitraging noise”.

Note that this example is just for educational purposes. I don't recommend you run this strategy live unless you understand it very well.

To achieve this we will use **Bollinger Bands**.

Bollinger Bands are price bands that are X standard deviations above and below a moving average of price.

Bollinger Bands

In our example we will use price that is above or below 2 standard deviations of the recent price average as the buy and sell signal, but in practice you want to pick a number of standard deviations that results in the price being within the Bollinger Bands **around 90-95% of the time**.

If the price too often hits or exceeds the Bollinger Bands, you may get many false signals.

Join over 10,000 future traders and get our **10-day "Know-What-To-Google" Algo Trading email course**. 1 short email a day for 10 days.

"I have not read it, but I'm sure it is great." - My girlfriend

Your first name

Your email address

Give me the 10-day crash course!

- **lookbackPeriod**: how far back to use to calculate to moving averages (in terms of the resolution)
- **standardDeviations**: the amount of standard deviations above and below the moving average to calculate the Bollinger Bands
- **movingAverageType**: the type of moving average to use (simple, exponential etc.)
- **Resolution**: the bar period to use for the price data (Minute, Hour, Daily etc.)

Implementation

Now let's have a go building our mean reversion strategy on QuantConnect!

Firstly, let's create our algorithm, inheriting as always from the **QCAgorithm** class, and setup our `Initialize()` function:

Python

```

1  import pandas as pd
2
3
4  class LeanHogsBollingerBandsAlgorithm(QCAgorithm):
5
6      def Initialize(self):
7
8          self.SetStartDate(2015, 1, 1)    #Set Start Date
9          self.SetEndDate(2020, 6, 1)      #Set End Date
10         self.SetCash(100000)              #Set Strategy Cash

```

Join over 10,000 future traders and get our **10-day "Know-What-To-Google" Algo Trading email course**. 1 short email a day for 10 days.

"I have not read it, but I'm sure it is great." - My girlfriend

Your first name

Your email address

Give me the 10-day crash course!

```
2 | self.contract = None
```

Then we need to subscribe to the lean hog's future chain data, which has accessor code `Futures.Meats.LeanHogs` :

```
1 | # Subscribe and set our expiry filter for the futures chain
2 | futureES = self.AddFuture(Futures.Meats.LeanHogs)
```

Python

Trading **futures** is a bit more complicated than trading normal equities.

Similar to *options*, futures contracts represent an agreement to buy or sell an asset at a future date at an agreed-upon price.

Unlike options however, the owner of the contract **must** buy or sell the asset at the agreed upon price on that future date, whereas options give the owner the right, but not the *obligation*, to do so.

As such, a base asset tends to have multiple futures contracts in circulation at any one time- each with different expiry dates for the future. So when you are trading futures, you must pick a specific contract (expiry date) to trade.

Subscribing to futures data does not give you normal price data, but rather a **FuturesChain**— a collection of information about the different contracts.

Join over 10,000 future traders and get our **10-day "Know-What-To-Google" Algo Trading email course**. 1 short email a day for 10 days.

"I have not read it, but I'm sure it is great." - My girlfriend

Your first name

Your email address

Give me the 10-day crash course!

Since here we have only subscribed to lean hog futures, there is only the lean hog futures chain in the FutureChain object.

In general, you can explore the future contract chain as such:

Python

```
1 # Explore the future contract chain
2 def OnData(self, slice):
3     for chain in slice.FutureChains.Values:
4         contracts = chain.Contracts
5         for contract in contracts.Values:
6             # do something with specific contract
```

A specific futures contract has the following properties:

Python

```
1 class FuturesContract:
2     self.Symbol # (Symbol) Symbol for contract needed to trade.
3     self.UnderlyingSymbol # (Symbol) Underlying futures asset.
4     self.Expiry # (datetime) When the future expires
5     self.OpenInterest # (decimal) Number of open interest.
6     self.LastPrice # (decimal) Last sale price.
7     self.Volume # (long) reported volume.
8     self.BidPrice # (decimal) bid quote price.
9     self.BidSize # (long) bid quote size.
10    self.AskPrice # (decimal) ask quote price.
11    self.AskSize # (long) ask quote size.
```

Join over 10,000 future traders and get our **10-day "Know-What-To-Google" Algo Trading email course**. 1 short email a day for 10 days.

"I have not read it, but I'm sure it is great." - My girlfriend

Your first name

Your email address

Give me the 10-day crash course!



Because of that, and the fact we want to give our trades time to play out before rotating to a new contract, we set a filter to subscribe to futures contracts expiring **no sooner than 30 days** in `Initialize()` .

We also set an upper-bound of 1080 days until expiry to define a time period of contract expiration we want to receive data for:

```
1 | futureES.SetFilter(TimeSpan.FromDays(30), TimeSpan.FromDays(1080))
```

Python

We are going to use four other functions besides `Initialize(self)` in algorithm. They will be:

- `OnData(self, slice)` – fires every time data is received (inbuilt in QuantConnect)
- `InitUpdateContract(self, slice)` needs to be called somewhere in our code (custom)
- `OnHour(self, sender, bar)` – fires every hour (inbuilt)
- `OnEndOfDay(self)` – fires at the end of every day (inbuilt)

Let's first get `OnEndOfDay()` out of the way, as we will simply use it to set our `new_day` boolean to **True** at the end of every day like so:

Join over 10,000 future traders and get our **10-day "Know-What-To-Google" Algo Trading email course**. 1 short email a day for 10 days.

"I have not read it, but I'm sure it is great." - My girlfriend

Your first name

Your email address

Give me the 10-day crash course!

need to be rolled over and update function (remember, we set our `self._day` to be True in `Initialize()`, so the first time this function is called we pass this check):

Python

```
1 def InitUpdateContract(self, slice):
2     # Reset daily – everyday we check whether futures need to be
3     rolled
4     if not self.new_day:
        return
```

Now we will perform a check to see if we are already trading a contract and if its expiry is at least 3 days away. If both these facts are true, we again skip the rest of the function.

Python

```
1 if self.contract != None and (self.contract.Expiry - self.Time).days
2 >= 3: # rolling 3 days before expiry
    return
```

If it is a new day, and if we do not have a contract selected or we do and it's within 3 days of expiry, we print in the backtest logs the name of the contract that's expiring and how long it has to expiration using the `self.Log("message")` method, and liquidate our current positions in preparation to trade a new contract:

Join over 10,000 future traders and get our **10-day "Know-What-To-Google" Algo Trading email course**. 1 short email a day for 10 days.

"I have not read it, but I'm sure it is great." - My girlfriend

Your first name

Your email address

Give me the 10-day crash course!

We then proceed to select a new contract first by getting a list of the contracts from the future contract chain:

Python

```
1 | # get list of contracts
2 | contracts = list(chain.Contracts.Values)
3 | chain_contracts = list(contracts) #[contract for contract in chain]
```

Then order the contracts by expiry date from newest to oldest:

Python

```
1 | # order list of contracts by expiry date: newest --> oldest
2 | chain_contracts = sorted(chain_contracts, key=lambda x: x.Expiry)
```

And then by picking out a contract early in that list (remember, we already set a filter so that none of these contracts expire within the next 30 days anyway):

Python

```
1 | # pick out contract and log contract name
2 | self.contract = chain_contracts[1]
3 | self.Log("Setting contract to:
    {}".format(self.contract.Symbol.Value))
```

Join over 10,000 future traders and get our **10-day "Know-What-To-Google" Algo Trading email course**. 1 short email a day for 10 days.

"I have not read it, but I'm sure it is great." - My girlfriend

Your first name

Your email address

Give me the 10-day crash course!

To keep things short and because you would normally have more flexible data resolution anyway when trading equities and cryptocurrencies etc., we will skip over exactly how consolidators work. But if you do need to build your own custom one for something else, you can learn about them in the QuantConnect [consolidating data docs](#)!

Now we have hourly data for our futures contracts, we will initialise a BollingerBand indicator object:

Python

```
1 | # Set up indicator
2 | self.Bolband = self.BB(self.contract.Symbol, 50, 2,
   | MovingAverageType.Simple, Resolution.Hour)
```

Here we set it to analyse price data from the contract we are trading- calculating a simple moving average on an hourly timeframe with the last 50 hours of data and creating bollinger bands ± 2 standard deviations from this moving average.

It can be hard to know what to set the sensitivity (standard deviations) to, but for curiosity's sake this is what weekly lean hog's future data looks like with Bollinger bands set at ± 1.5 standard deviations:

Join over 10,000 future traders and get our **10-day "Know-What-To-Google" Algo Trading email course**. 1 short email a day for 10 days.

"I have not read it, but I'm sure it is great." - My girlfriend

Your first name

Your email address

Give me the 10-day crash course!

```
-
4 | for bar in history.itertuples():
5 |     if bar.time.minute == 0 and ((self.Time-
    bar.time)/pd.Timedelta(minutes=1)) >= 2:
        self.Bolband.Update(bar.time, bar.close)
```

We now set `self.new_day` to **False** because we have already rolled-forward the contract we are trading on this day, concluding our `InitUpdateContract()` function:

Python

```
1 | self.new_day = False
```

We will now deal with our `OnData()` function.

This activates whenever new data becomes available for the FuturesChains we have subscribed to.

As such, whenever this function fires, we want to run `InitUpdateContract()` to check the status of our activate contract, roll over to a new contract if necessary (or pick an initial one at the start of the algorithm), liquidate old positions and warm up a new Bollingerband indicator:

Python

```
1 | def OnData(self, slice):
2 |     self.InitUpdateContract(slice)
```

Join over 10,000 future traders and get our **10-day "Know-What-To-Google" Algo Trading email course**. 1 short email a day for 10 days.

"I have not read it, but I'm sure it is great." - My girlfriend

Your first name

Your email address

Give me the 10-day crash course!

the futures contract:

Python

```
1 | def OnHour(self, sender, bar):
2 |     if (self.Bolband != None and self.Bolband.IsReady):
3 |         if bar.Symbol == self.contract.Symbol:
4 |             price = bar.Close
```

Continuing within the previous if statement, we will check the number of contracts we own for the current contract we are trading via

`self.Portfolio[symbol].Quantity :`

Python

```
1 | holdings = self.Portfolio[self.contract.Symbol].Quantity
```

Go [here](#) to learn a bit more about the Portfolio object and what else you can do with it!

We will now finally implement our trading logic!

If we do not currently own any amount the contract we are trading and the price for the contract dips below the lower Bollinger band, we will market buy 2 contracts:

Join over 10,000 future traders and get our **10-day "Know-What-To-Google" Algo Trading email course**. 1 short email a day for 10 days.

"I have not read it, but I'm sure it is great." - My girlfriend

Your first name

Your email address

Give me the 10-day crash course!

Python

```
1 | # sell if price closes above the upper bollinger band
2 | if holdings > 0 and price > self.Bolband.UpperBand.Current.Value:
3 |     self.Log("SELL >> {}".format(price))
4 |     self.Liquidate()
```

And that's how we execute a mean reversion strategy on futures!

Just to finish off the function, we will plot our Bollinger bands as the backtest executes, to get a nice visual check that things are working as inspected:

Python

```
1 | self.Plot("BB", "MiddleBand", self.Bolband.MiddleBand.Current.Value)
2 | self.Plot("BB", "UpperBand", self.Bolband.UpperBand.Current.Value)
3 | self.Plot("BB", "LowerBand", self.Bolband.LowerBand.Current.Value)
```

The format here is `Plot("name of chart to plot in", "name of specific line in chart", valueToPlot)`.

The function finishes on the outside if level with an else statement to print to the logs that the Bollinger bands have not yet finished warming up if that is the case:

Python

```
1 | else:
2 |     self.Log('Bollinger Bands not ready yet')
```


Join over 10,000 future traders and get our **10-day "Know-What-To-Google" Algo Trading email course**. 1 short email a day for 10 days.

"I have not read it, but I'm sure it is great." - My girlfriend

Your first name

Your email address

Give me the 10-day crash course!

Join over 10,000 future traders and get our **10-day "Know-What-To-Google" Algo Trading email course**. 1 short email a day for 10 days.

"I have not read it, but I'm sure it is great." - My girlfriend

Your first name

Your email address

Give me the 10-day crash course!

```

13         self.contract = None
14
15
16         # Subscribe and set our expiry filter for the futures chain
17         futureES = self.AddFuture(Futures.Meats.LeanHogs)
18         futureES.SetFilter(TimeSpan.FromDays(30),
19         TimeSpan.FromDays(720))
20
21
22         def OnData(self, slice):
23
24             self.InitUpdateContract(slice)
25
26             def InitUpdateContract(self, slice):
27                 # Reset daily - everyday we check whether futures need to be
28                 rolled
29                 if not self.new_day:
30                     return
31
32                 if self.contract != None and (self.contract.Expiry -
33                 self.Time).days >= 3: # rolling 3 days before expiry
34                     return
35
36                 for chain in slice.FutureChains.Values:
37                     # If we trading a contract, send to log how many days
38                     until the contract's expiry
39                     if self.contract != None:
40                         self.Log('Expiry days away {} -
41                         {}'.format((self.contract.Expiry-self.Time).days,
42                         self.contract.Expiry))
43
44                     # Reset any open positions based on a contract
45                     rollover.
46                     self.Log('RESET: closing all positions')
47                     self.Liquidate()
48

```

Join over 10,000 future traders and get our **10-day "Know-What-To-Google" Algo Trading email course**. 1 short email a day for 10 days.

"I have not read it, but I'm sure it is great." - My girlfriend

Your first name

Your email address

Give me the 10-day crash course!

```

63         # Set up consolidators.
64         one_hour =
65         TradeBarConsolidator(TimeSpan.FromMinutes(60))
66         one_hour.DataConsolidated += self.OnHour
67
68
69         self.SubscriptionManager.AddConsolidator(self.contract.Symbol,
70         one_hour)
71
72         # Set up indicators
73         self.Bolband = self.BB(self.contract.Symbol, 50, 2,
74         MovingAverageType.Simple, Resolution.Hour)
75
76
77         history = self.History(self.contract.Symbol, 50*60,
78         Resolution.Minute).reset_index(drop=False)
79
80         for bar in history.itertuples():
81             if bar.time.minute == 0 and ((self.Time-
82             bar.time)/pd.Timedelta(minutes=1)) >= 2:
83                 self.Bolband.Update(bar.time, bar.close)
84
85         self.new_day = False
86
87
88         def OnHour(self, sender, bar):
89
90             if (self.Bolband != None and self.Bolband.IsReady):
91                 if bar.Symbol == self.contract.Symbol:
92                     price = bar.Close
93
94                     holdings =
95                     self.Portfolio[self.contract.Symbol].Quantity
96
97                     # buy if price closes below lower bollinger band

```

Join over 10,000 future traders and get our **10-day "Know-What-To-Google" Algo Trading email course**. 1 short email a day for 10 days.

"I have not read it, but I'm sure it is great." - My girlfriend

Your first name

Your email address

Give me the 10-day crash course!

```

        self.Bollband.UpperBand.Current.Value),
        self.Plot("BB", "UpperBand",
self.Bolband.UpperBand.Current.Value)
        self.Plot("BB", "LowerBand",
self.Bolband.LowerBand.Current.Value)

    else:
        self.Log('Bollinger Bands not ready yet')

def OnEndOfDay(self):
    self.new_day = True

```

Now we can proceed to the most exciting bit of all- the backtest!

Hit the **Backtest** button and the backtest should begin initialising.

It will take a few minutes for this particular backtest to complete- you can either watch the graphs and metrics update in real time, or just come back when its complete!

*results
of our
backtest!*

As you can see our strategy had some ups and down, but we actually finished with a 5.62% return- not bad without much calibration!

Join over 10,000 future traders and get our **10-day "Know-What-To-Google" Algo Trading email course**. 1 short email a day for 10 days.

"I have not read it, but I'm sure it is great." - My girlfriend

Your first name

Your email address

Give me the 10-day crash course!

effectively our own custom plotted ~~pseudo-benchmark~~ — we can see we definitely appeared to do better than buying and holding!

Finally if you scroll down a bit, you will find a bunch of detailed statistics about the back test:

Here is our backtest Orders history in action:

*Orders
history*

And the Logs history:

*Logs
history*

What is QuantConnect's Strategy Development Framework (SDF)?

We just built an algorithm without using any of QuantConnect's SDF modules.

We will now build an algorithm using the SDF instead.

Join over 10,000 future traders and get our **10-day "Know-What-To-Google" Algo Trading email course**. 1 short email a day for 10 days.

"I have not read it, but I'm sure it is great." - My girlfriend

Your first name

Your email address

Give me the 10-day crash course!

Every module must pass on an object in a specific format to the next module in the chain for the algorithm to function automatically and correctly.

This can all be a lot to get your head around at first, so if you get stuck you can read the SDF docs [here](#) and/or work through [The Algorithm Framework bootcamp](#).

Let's begin with an example of using a Universe Selection module!

Step 1: Using QuantConnect's Universe Selection example: Tech stocks

Firstly, click on "Create New Algorithm".

Then scroll down to the **Universe** section and explore the range of pre-made universes available.

By clicking on a module, you will be given a description of how that module functions and what conditions it filters by.

Note that the SDF modules on offer vary slightly depending on whether you have Python or C# selected as the language.

We are going to go with the "Technology stocks" module.

Join over 10,000 future traders and get our **10-day "Know-What-To-Google" Algo Trading email course**. 1 short email a day for 10 days.

"I have not read it, but I'm sure it is great." - My girlfriend

Your first name

Your email address

Give me the 10-day crash course!



There's a lot going on that is hard to follow unless you know a lot about the intricacies about how QuantConnect's algorithms are set up under the hood, but that's okay- the module works according to its description without any intervention required.

The key point is that the module filters stocks down by all the various conditions described in the code (which you are free to try adjusting if you would like!) and returns a list of stock symbols to trade to the Alpha Generation module.

Step 2: Using QuantConnect's Alpha Creation example: Smart Insider

Overview

The Alpha creation module is perhaps the section that will require the most manual work out of all the SDF modules since many of the other modules function automatically with just a few input parameters, however often we will need to interpret the data fed to us in an Alpha Creation module, write trading logic and generate **insights** to our satisfaction to pass forward to the Portfolio Construction module.

That said this isn't always true- for instance some indicators like the RSI module will generate insights without further work, but be careful because some of these modules do not assign insight weights- which are needed by some Portfolio

Join over 10,000 future traders and get our **10-day "Know-What-To-Google" Algo Trading email course**. 1 short email a day for 10 days.

"I have not read it, but I'm sure it is great." - My girlfriend

Your first name

Your email address

Give me the 10-day crash course!



This is what the Smart Insider module code looks like to begin with:

Lets first deal with the `OnSecuritiesChanged(self, algorithm, changes)` method at the bottom.

Its purpose is to subscribe to transaction and intention data for any new stocks entering our Tech Stock universe, and unsubscribe to that data for any stocks leaving (remember our universe continuously updates such that all stocks in the universe meet the criteria we set).

It is almost good as is, however we are going to add one extra line–
`algorithm.Liquidate(security.Symbol)` – to make sure we liquidate our positions in any stocks that are removed from our universe.

The second part of our `OnSecuritiesChanged()` method now looks like this:

Join over 10,000 future traders and get our **10-day "Know-What-To-Google" Algo Trading email course**. 1 short email a day for 10 days.

"I have not read it, but I'm sure it is great." - My girlfriend

Your first name

Your email address

Give me the 10-day crash course!



Now lets take look at the main `Update(self, algorithm, data)` method.

Generating Insights

The critical part is that every Alpha module must return **Insights**.

The Update method returns an array of Insight objects.

An Insight is a *single* prediction for an asset.

These can be thought of as actionable trading signals, indicating the asset direction, magnitude, and confidence in the near future. All insights can take a weight parameter to set the desired weighting for the insight.

Insights have the following properties:

We can also optionally bring the price or volatility property onto the outside and create an insight object like so:

Python

```
1 | # Skipping magnitude, confidence and source model and assigning 25%  
2 | to weighting.  
   | insight = Insight.Price("IBM", timedelta(minutes = 20),  
   | InsightDirection.Up, None, None, None, 0.25)
```

Join over 10,000 future traders and get our **10-day "Know-What-To-Google" Algo Trading email course**. 1 short email a day for 10 days.

"I have not read it, but I'm sure it is great." - My girlfriend

Your first name

Your email address

Give me the 10-day crash course!

As you can see we will need to use the **intention** and **transaction** properties to create some trading logic and create **insights**.

For our demonstration here we will go for a very basic strategy: we will buy shares of a stock when a company announces an **intention** of a stock buyback, hoping for the buyback to increase the price through temporary increased buying pressure, and sell the stock when a **transaction** event confirms the completion of the buy back (hopefully for more!).

So lets do that- whenever an intention event occurs we will create a **buy insight** (`InsightDirection.Up`) and append it to the list of insights we will return:

Python

```
1 # Iterate over transactions and parse information
2 for intention in intentions.Values:
3     ## Generate Insights!
4     # Skipping magnitude, confidence and source model and assigning
5     25% to weighting.
6     insight = Insight.Price(intention.Symbol.Underlying,
        timedelta(days = 5), InsightDirection.Up, None, None, None, 0.25)
        insights.append(insight)
```

Here `intention.Symbol.Underlying` is how we access the symbol for the stock whose intention we are analysing, and we've given the insight a valid period of 5 days and a weighting of 0.25 (again weighting isn't always necessary, but we'll need it for the Portfolio Construction module we will use).

Join over 10,000 future traders and get our **10-day "Know-What-To-Google" Algo Trading email course**. 1 short email a day for 10 days.

"I have not read it, but I'm sure it is great." - My girlfriend

Your first name

Your email address

Give me the 10-day crash course!

Python

```
1 # Iterate over transactions and parse information
2 for transaction in transactions.Values:
3     ## Generate Insights!
4     # Skipping magnitude, confidence and source model and assigning
5     25% to weighting.
6     if transaction.VolumePercentage != None and
7     transaction.VolumePercentage > 5:
8         insight = Insight.Price(transaction.Symbol.Underlying,
9                                 timedelta(days = 5), InsightDirection.Down, None, None, None, 0.25)
10        insights.append(insight)
```

All together our adjusted **SmartInsiderAlphaModel.py** module now looks like this:

Join over 10,000 future traders and get our **10-day "Know-What-To-Google" Algo Trading email course**. 1 short email a day for 10 days.

"I have not read it, but I'm sure it is great." - My girlfriend

Your first name

Your email address

Give me the 10-day crash course!

```

13 repurchases
14     ## its own stock. It reduces the number of shares available
15 to
16     ## other investors, which in theory should reduce the supply
17 of
18     ## shares and increase the stock price.
19
20     ## Smart Insider has two data sets available to use in your
21 algorithm.
22     ## The Intentions data set is an announcement that
23 establishes the intention
24     ## to buy-back shares of the company. When the buy-back
25 occurs this triggers
26     ## a Transaction event with details about the execution of
27 the buyback.
28     ## Intention events always come before the Transaction
29 event.
30
31     # Fetch all transactions and intentions
32     intentions = data.Get(SmartInsiderIntention)
33     transactions = data.Get(SmartInsiderTransaction)
34
35     # Iterate over transactions and parse information
36     for intention in intentions.Values:
37         ## Generate Insights!
38         # Skipping magnitude, confidence and source model and
39 assigning 25% to weighting.
40         insight = Insight.Price(intention.Symbol.Underlying,
41 timedelta(days = 5), InsightDirection.Up, None, None, None, 0.25)
42         insights.append(insight)
43
44     # Iterate over transactions and parse information
45     for transaction in transactions.Values:
46         ## Generate Insights!
47         # Skipping magnitude, confidence and source model and
48 assigning 25% to weighting.
```

Join over 10,000 future traders and get our **10-day "Know-What-To-Google" Algo Trading email course**. 1 short email a day for 10 days.

"I have not read it, but I'm sure it is great." - My girlfriend

Your first name

Your email address

Give me the 10-day crash course!

```

new equity
    for security in changes.AddedSecurities:
        if security.Type == SecurityType.Equity:
            transaction =
algorithm.AddData(SmartInsiderTransaction, security.Symbol).Symbol
            intention = algorithm.AddData(SmartInsiderIntention,
security.Symbol).Symbol
            self.altDataSymbols[security.Symbol] = (transaction,
intention)

    ## Remove SmartInsider Transaction and Intention data for
each new equity
    for security in changes.RemovedSecurities:
        if security.Type == SecurityType.Equity:
            algorithm.Liquidate(security.Symbol)
            transaction, intention =
self.altDataSymbols.pop(security.Symbol, (None, None))
            algorithm.RemoveSecurity(transaction) if transaction
is not None else None
            algorithm.RemoveSecurity(intention) if transaction
is not None else None

```

Step 3: Using QuantConnect's Portfolio Construction example: Insight Weighted

Now lets pick a Portfolio Construction module.

Thankfully, these all perform everything under the hood and do not require manual intervention, however as we've already mentioned some do require **insight weights** to be emitted in the insight objects to function.

Join over 10,000 future traders and get our **10-day "Know-What-To-Google" Algo Trading email course**. 1 short email a day for 10 days.

"I have not read it, but I'm sure it is great." - My girlfriend

Your first name

Your email address

Give me the 10-day crash course!

To continue our example we will go with the **Insight Weighted** module:

This simply uses **Insight Weight** we assign to determine the holding size as by the description.

Step 4: Using QuantConnect's Execution Engine example: Immediate

As with Portfolio Construction, barely extra work for us here. Simply pick one of the three choices after giving their descriptions a thorough read:

- **Immediate**
- **Standard Deviation**
- **VWAP** (Volume Weighted Average Price)

These all determine what conditions are necessary to place trades to reach our Portfolio Construction target goals.

For instance the **Standard Deviation** module takes advantage of mean-reversion and only looks to place buy orders when the price is below X standard deviations of a moving average.

It is the only Execution module that requires additional information- you need to pass it a look-back period, standard deviation setting, and data resolution much like our previous Bollinger bands:

Join over 10,000 future traders and get our **10-day "Know-What-To-Google" Algo Trading email course**. 1 short email a day for 10 days.

"I have not read it, but I'm sure it is great." - My girlfriend

Your first name

Your email address

Give me the 10-day crash course!

Finally lets pick a Risk Management module.

These all just determine conditions of poor performance to liquidate your portfolio (or part of your portfolio) under.

The choices are:

- **Maximum Unrealized Profit**
- **Trailing Stop Drawdown**
- **Portfolio Drawdown**
- **Sector Exposure**
- **Maximum Drawdown**

Again, just give their **descriptions** a good read.

These at most require passing a decimal in the constructor indicating the maximum acceptable amount by whatever metric before action is taken.

For our example lets go with the Portfolio Drawdown module:

```
1 | self.SetRiskManagement(MaximumDrawdownPercentPortfolio(0.03))
```

Python

This liquidates our portfolio if the greatest peak to trough portfolio value loss is more than 3%.

Join over 10,000 future traders and get our **10-day "Know-What-To-Google" Algo Trading email course**. 1 short email a day for 10 days.

"I have not read it, but I'm sure it is great." - My girlfriend

Your first name

Your email address

Give me the 10-day crash course!

Outside of that we only made alterations to our **SmartInsiderAlphaModel.py**:
(scroll back up a few sections for the full code).

Let's finally hit that backtest button!

Not that terrible- at least we didn't lose money!

But we would have done better just buying and holding...

Check out our Insights in action!

Final Thoughts

So there you have it- a basic guide on QuantConnect!

By this point it's probably self evident that the biggest drawback to QuantConnect is the huge amount of stuff you have to learn.

The documentation to do so is reasonable but we personally found the example algorithms QuantConnect provides to sometimes be unnecessarily complex and poorly commented- making it difficult to follow where stuff is coming from if you are not already adept.

Join over 10,000 future traders and get our **10-day "Know-What-To-Google" Algo Trading email course**. 1 short email a day for 10 days.

"I have not read it, but I'm sure it is great." - My girlfriend

Your first name

Your email address

Give me the 10-day crash course!

LINK TO DOWNLOAD CODE

You can find the code used in this article [here](#).

Get our 10-day "Know-What-To-Google" Algo Trading email course.

1 short email a day for 10 days.

Over 10,000 future traders have taken this email course.

"I have not read it, but I'm sure it is great." - My girlfriend

Join over 10,000 future traders and get our **10-day "Know-What-To-Google" Algo Trading email course**. 1 short email a day for 10 days.

"I have not read it, but I'm sure it is great." - My girlfriend

Your first name

Your email address

Give me the 10-day crash course!

Greg Blann



Programming Trading