《zipline.v0.7 函数大全·zw 汉化注解版》

zw 量化开源·系列课件

zw 开源量化团队 QQ 群：533233771。

作者：zw=智王+字王

2016.01.20

Zipline

Zipline is a Pythonic algorithmic trading library. It is an event-driven system that supports both backtesting and live-trading.

Zipline is currently used in production as the backtesting and live-trading engine powering Quantopian – a free, community-centered, hosted platform for building and executing trading strategies.

zw量化交易·实盘操作
魔鬼训练营
THE DEVIL TRAINING CAMP
Win or Home
要么全赢，要么滚蛋
国内首个·量化实盘·魔鬼训练营
zw量化实盘·免费公开课
主讲：字王　课时：约90分钟
时间预告：2016年1月10日 20点
QQ群：124134140(zwPython量化交易&足彩大数据)
博客：http://blog.sina.com.cn/zbrow
网站：http://www.ziwang.com

- 字王 Git 项目总览：github.com/ziwang-com/，
  包括：字王 4k 云字库，zwPython、zwpy_lst
- Zw 量化 QQ 群：124134140（AI 量化，足彩大数据、云字库、zwPython）
- zw 开源量化团队 QQ 群：533233771。
- 技术 Blog：blog.sina.com.cn/zbrow （AI 量化、足彩大数据、字库）
- www.cnblogs.com/ziwang/ （机器视觉）
- 网盘下载：http://pan.baidu.com/s/1bnSqTxd
- zw 网站：http://www.ziwang.com

github，全球最大的极客创意平台，欢迎大家参与

前言

(zipline 与 LAPACK）

Zipline 安装

Zipline 是目前比较热门的量化交易算法包，网络上的介绍是：

Zipline 是一个交易算法库，该系统是对现场交易系统如何运转的一个近似，可以对历史数据进行投资算法的回溯检验。

Zipline 目前作为 Quantopian 的回溯检验引擎。

特性

● 使用简单
● 包括常用统计方法如移动平均和线性回归
● 与现有 python 生态圈能很好融合
● 一些常用统计和机器学习库，如 matplotlib、scipy、statsmodels 和 sklearn，支持交易系统的开发、数据分析和可视化

不过，zipline 的安装，却非常麻烦。

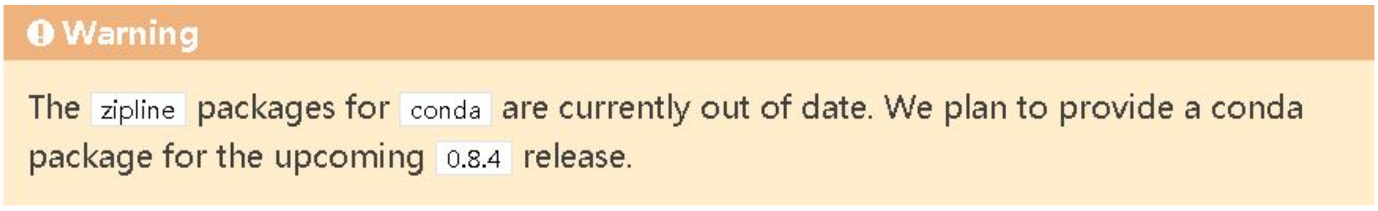
主要是因为，zipline 0.8 后，为了速度优化，使用了 LAPACK 算法库。（这个后面细谈）

Zipline 最新版本是 v0.83，可是，无论怎么安装，用 winPy 的安装程序，还是 pip，都不行。

原本以为是版本冲突，于是降级安装，py3.4、py2.7，都不行。

转到 zipline 网站：http://www.zipline.io

可能是，因为安装吐槽的人，太多，官网 install 页码，80%都在回答各种平台，各种安装模式，最后，官方声明：v0.83 的确不好安装，大家等下一个版本： v0.84，好不好。


Warning

The zipline packages for conda are currently out of date. We plan to provide a conda package for the upcoming 0.8.4 release.

居然 py 降级安装不行，那就 zipline 降级安装。

一路测试：

■ v0.82，不行；
■ v0.80，更加不行，居然只有 mac 版本，可能开发团队用的是马刺，Linux 之类的。
■ v0.7，居然 ok 了，py2.7，py3.4，py3.5，全部 ok；

看来，原因，还是出在 LAPACK 上面。

开发团队可能用的是 linux 平台，当然，实事求是而言，大家用多了 python，以后工作中，建议还是在 linux 环境下，使用 python。

因为 py 和大部分开源软件样，许多非 python 原生的模块库，都是一源码方式存在的，主要是 c，linux 本身就内置了 gcc 开源编译环境，集成、编译、安装非常方便。

想 LAPACK 模块库，在 linux 下安装，直接 install，就 ok 了。

LAPACK 是什么？

LAPACK 是由美国国家科学基金等资助开发的著名公开软件。

LAPACK 包含了求解科学与工程计算中最常见的数值线性代数问题，如求解线性方程组、线性最小二乘问题、特征值问题和奇异值问题等[1] 。

为什么，LAPACK 的安装这么麻烦呢？

因为，LAPACK 算法库，源码是 fotran 的，大家没看错，是 fotran 版本。

因为，fotran 比 c 版本快 20%，然后，用 c 封装了个 c-api 接口，给各个软件用。

不过这个 lapack 因为是开源版本，速度比 intel 商业加速算法包 mkl，里面的 lapack 还是要慢不少的。

这个 LAPACK 的配置，安装很啰嗦，类似 cuda，而且，初学者，一般用不上。

日后到了工作环境，如果，需要优化速度，有以下几种解决方案：

- 使用 linux 环境，安装各种加速库
- 使用 cuda、opencl、numba 等各种 gpu 加速方案，这个是终极方案，可提速数百倍。
- 使用其他的 python 原生量化交易算法模块库，这方面的软件包还是不少的。
-

更多请浏览 zw 网站: http://ziwang.com

或技术 blog:http://blog.sina.com.cn/zbrow

ZW 量化开源团队简介

zw 开源量化团队 QQ 群号：　533233771

zw 开源量化团队，英文名称暂定：zwQTT：ziwang.com Quant Tearm，

zw 开源量化团队，是个免费的开源公益组织，专业从事国内外最新的金融、量化资料、软件、教程等各种相关资源的引进、翻译、宣传、托管。

同时，在能力所及的范围内，进行相关的量化开源软件开发。

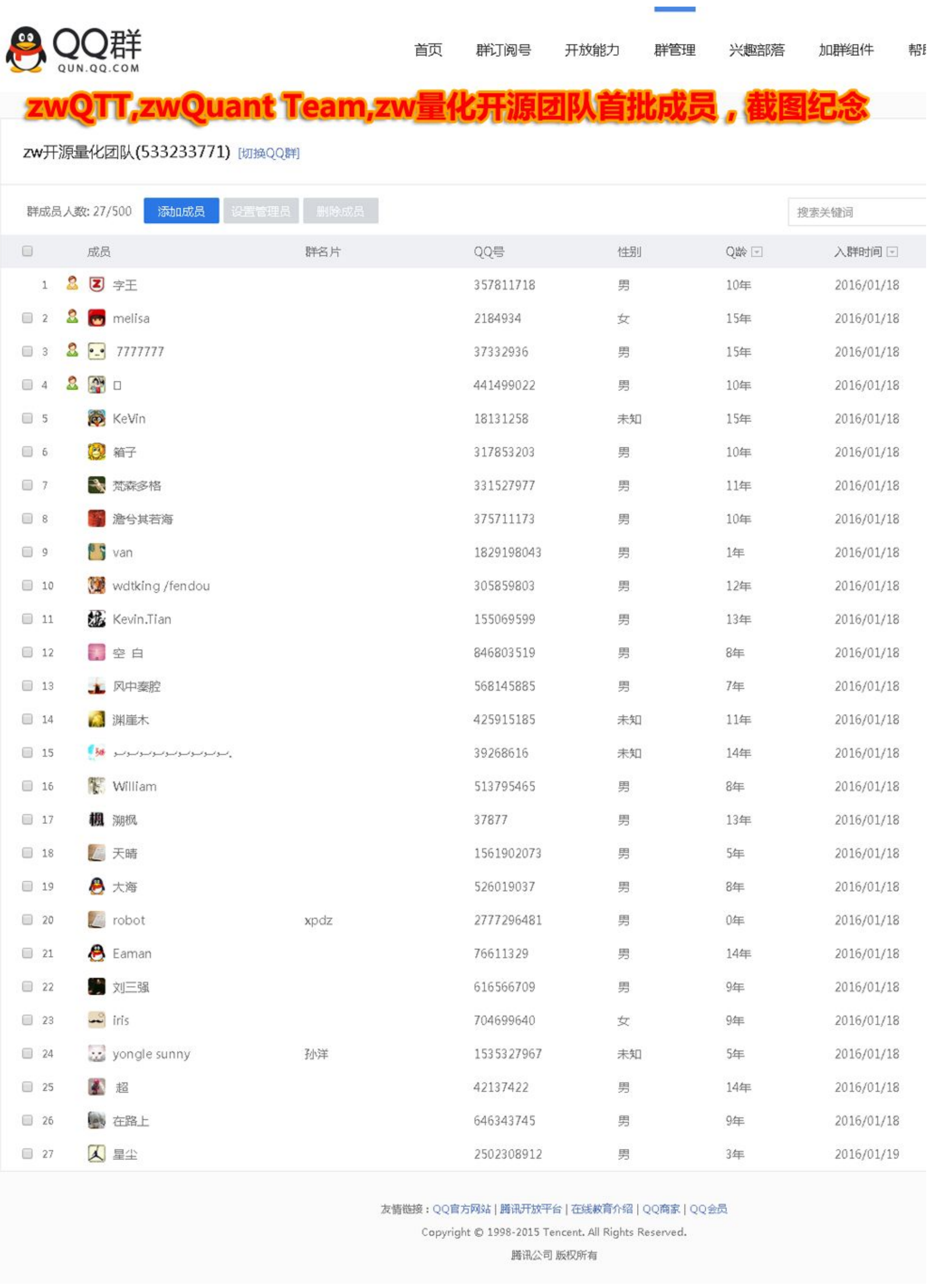
有关团队介绍，可参见：

《zw 开源量化团队・成立纪念》 http://ziwang.com/?p=214
《zw 开源量化团队・约法三章》 http://ziwang.com/?p=212

新人申请，请先浏览，以上文档，填写表格，再联系团队管理员。

更多请浏览 zw 网站:http://ziwang.com
或技术 blog:http://blog.sina.com.cn/zbrow

附图：是 zw 开源量化团队，QQ 群首批成员截图纪念



zwQTT,zwQuant Team,zw量化开源团队首批成员，截图纪念

zw开源量化团队(533233771) [切换QQ群]

群成员人数: 27/500 添加成员 设置管理员 删除成员 搜索关键词

	成员	群名片	QQ号	性别	Q龄	入群时间
1	字王		357811718	男	10年	2016/01/18
2	melisa		2184934	女	15年	2016/01/18
3	7777777		37332936	男	15年	2016/01/18
4	□		441499022	男	10年	2016/01/18
5	KeVin		18131258	未知	15年	2016/01/18
6	箱子		317853203	男	10年	2016/01/18
7	梵森多格		331527977	男	11年	2016/01/18
8	澹兮其若海		375711173	男	10年	2016/01/18
9	van		1829198043	男	1年	2016/01/18
10	wdtking /fendou		305859803	男	12年	2016/01/18
11	Kevin.Tian		155069599	男	13年	2016/01/18
12	空 白		846803519	男	8年	2016/01/18
13	风中秦腔		568145885	男	7年	2016/01/18
14	渊崖木		425915185	未知	11年	2016/01/18
15	[illegible]		39268616	未知	14年	2016/01/18
16	William		513795465	男	8年	2016/01/18
17	溯枫		37877	男	13年	2016/01/18
18	天晴		1561902073	男	5年	2016/01/18
19	大海		526019037	男	8年	2016/01/18
20	robot	xpdz	2777296481	男	0年	2016/01/18
21	Eaman		76611329	男	14年	2016/01/18
22	刘三强		616566709	男	9年	2016/01/18
23	iris		704699640	女	9年	2016/01/18
24	yongle sunny	孙洋	1535327967	未知	5年	2016/01/18
25	超		42137422	男	14年	2016/01/18
26	在路上		646343745	男	9年	2016/01/18
27	星尘		2502308912	男	3年	2016/01/19

友情链接：QQ官方网站 | 腾讯开放平台 | 在线教育介绍 | QQ商家 | QQ会员

Copyright © 1998-2015 Tencent. All Rights Reserved.

腾讯公司 版权所有

TradingAlgorithm

TradingAlgorithm 所在模块： zipline.algorithm:

类定义：TradingAlgorithm(builtins.object)

```
|   Base class for trading algorithms. Inherit and overload
|   initialize() and handle_data(data).
|
|   A new algorithm could look like this:
|   ```
|   from zipline.api import order
|
|   def initialize(context):
|       context.sid = 'AAPL'
|       context.amount = 100
|
|   def handle_data(self, data):
|       sid = context.sid
|       amount = context.amount
|       order(sid, amount)
|   ```
|   To then to run this algorithm pass these functions to
|   TradingAlgorithm:
|
|   my_algo = TradingAlgorithm(initialize, handle_data)
|   stats = my_algo.run(data)
|
|  【类方法定义】
|
|   __init__(self, *args, **kwargs)
|       Initialize sids and other state variables.
|
|      【参数】
|       :Optional:
|           initialize : function
|               Function that is called with a single
|               argument at the begninning of the simulation.
|           handle_data : function
|               Function that is called with 2 arguments
|               (context and data) on every bar.
|           script : str
|               Algoscript that contains initialize and
|               handle_data function definition.
```

```
|               data_frequency : str (daily, hourly or minutely)
|                   The duration of the bars.
|               capital_base : float <default: 1.0e5>
|                   How much capital to start with.
|               instant_fill : bool <default: False>
|                   Whether to fill orders immediately or on next bar.
|
|  __repr__(self)
|       N.B. this does not yet represent a string that can be used
|       to instantiate an exact copy of an algorithm.
|
|       However, it is getting close, and provides some value as something
|       that can be inspected interactively.
|
|  add_history(self, bar_count, frequency, field, ffill=True)
|
|  add_transform(self, transform_class, tag, *args, **kwargs)
|       Add a single-sid, sequential transform to the model.
|
|       【参数】
|               transform_class : class
|                   Which transform to use. E.g. mavg.
|               tag : str
|                   How to name the transform. Can later be access via:
|                   data[sid].tag()
|
|       Extra args and kwargs will be forwarded to the transform
|       instantiation.
|
|  analyze(self, perf)
|
|  cancel_order(self, order_param)
|
|  get_datetime(self)
|       Returns a copy of the datetime.
|
|  get_generator(self)
|       Override this method to add new logic to the construction
|       of the generator. Overrides can use the _create_generator
|       method to get a standard construction generator.
|
|  get_open_orders(self, sid=None)
|
|  get_order(self, order_id)
|
|  handle_data(self, data)
```

```
|
|  history(self, bar_count, frequency, field, ffill=True)
|
|  initialize(self, *args, **kwargs)
|        Call self._initialize with `self` made available to Zipline API
|        functions.
|
|  on_dt_changed(self, dt)
|        Callback triggered by the simulation loop whenever the current dt
|        changes.
|
|        Any logic that should happen exactly once at the start of each datetime
|        group should happen here.
|
|  order(self, sid, amount, limit_price=None, stop_price=None, style=None)
|        Place an order using the specified parameters.
|
|  order_percent(self, sid, percent, limit_price=None, stop_price=None, style=None)
|        Place an order in the specified security corresponding to the given
|        percent of the current portfolio value.
|
|        Note that percent must expressed as a decimal (0.50 means 50\%).
|
|  order_target(self, sid, target, limit_price=None, stop_price=None, style=None)
|        Place an order to adjust a position to a target number of shares. If
|        the position doesn't already exist, this is equivalent to placing a new
|        order. If the position does exist, this is equivalent to placing an
|        order for the difference between the target number of shares and the
|        current number of shares.
|
|  order_target_percent(self, sid, target, limit_price=None, stop_price=None, style=None)
|        Place an order to adjust a position to a target percent of the
|        current portfolio value. If the position doesn't already exist, this is
|        equivalent to placing a new order. If the position does exist, this is
|        equivalent to placing an order for the difference between the target
|        percent and the current percent.
|
|        Note that target must expressed as a decimal (0.50 means 50\%).
|
|  order_target_value(self, sid, target, limit_price=None, stop_price=None, style=None)
|        Place an order to adjust a position to a target value. If
|        the position doesn't already exist, this is equivalent to placing a new
|        order. If the position does exist, this is equivalent to placing an
|        order for the difference between the target value and the
|        current value.
|
|  order_value(self, sid, value, limit_price=None, stop_price=None, style=None)
|        Place an order by desired value rather than desired number of shares.
|        If the requested sid is found in the universe, the requested value is
|        divided by its price to imply the number of shares to transact.
|
|        value > 0 :: Buy/Cover
|        value < 0 :: Sell/Short
|        Market order:       order(sid, value)
|        Limit order:        order(sid, value, limit_price)
|        Stop order:         order(sid, value, None, stop_price)
|        StopLimit order: order(sid, value, limit_price, stop_price)
|
|  raw_orders(self)
|        Returns the current open orders from the blotter.
|
|        N.B. this is not a property, so that the function can be passed
|        and called back from within a source.
|
|  raw_positions(self)
|        Returns the current portfolio for the algorithm.
|
|        N.B. this is not done as a property, so that the function can be
|        passed and called from within a source.
|
|  record(self, *args, **kwargs)
|        Track and record local variable (i.e. attributes) each day.
|
|  register_trading_control(self, control)
|        Register a new TradingControl to be checked prior to order calls.
|
|  run(self, source, overwrite_sim_params=True, benchmark_return_source=None)
|        Run the algorithm.
|
|       【参数】
|            source : can be either:
|                      - pandas.DataFrame
|                      - zipline source
|                      - list of sources
|
|               If pandas.DataFrame is provided, it must have the
|               following structure:
|               * column names must consist of ints representing the
|                 different sids
|               * index must be DatetimeIndex
|               * array contents should be price info.
|
|        【返回值】
|            daily_stats : pandas.DataFrame
|              Daily performance metrics such as returns, alpha etc.
|
|  set_commission(self, commission)
|
|  set_logger(self, logger)
|
|  set_long_only(self)
|      Set a rule specifying that this algorithm cannot take short positions.
|
|  set_max_order_count(self, max_count)
|      Set a limit on the number of orders that can be placed within the given
|      time interval.
|
|  set_max_order_size(self, sid=None, max_shares=None, max_notional=None)
|      Set a limit on the number of shares and/or dollar value of any single
|      order placed for sid.  Limits are treated as absolute values and are
|      enforced at the time that the algo attempts to place an order for sid.
|
|      If an algorithm attempts to place an order that would result in
|      exceeding one of these limits, raise a TradingControlException.
|
|  set_max_position_size(self, sid=None, max_shares=None, max_notional=None)
|      Set a limit on the number of shares and/or dollar value held for the
|      given sid. Limits are treated as absolute values and are enforced at
|      the time that the algo attempts to place an order for sid. This means
|      that it's possible to end up with more than the max number of shares
|      due to splits/dividends, and more than the max notional due to price
|      improvement.
|
|      If an algorithm attempts to place an order that would result in
|      increasing the absolute value of shares/dollar value exceeding one of
|      these limits, raise a TradingControlException.
|
|  set_slippage(self, slippage)
|
|  set_sources(self, sources)
|
|  set_transact(self, transact)
|      Set the method that will be called to create a
|      transaction from open orders and trade events.
|
|  set_transforms(self, transforms)
|
|  update_dividends(self, dividend_frame)
```

```
 |         Set DataFrame used to process dividends.    DataFrame columns should
 |         contain at least the entries in zp.DIVIDEND_FIELDS.
 |
 |  updated_portfolio(self)
 |
 |  validate_order_params(self, sid, amount, limit_price, stop_price, style)
 |         Helper method for validating parameters to the order API function.
 |
 |         Raises an UnsupportedOrderParameters if invalid arguments are found.
 |
 |  ----------------------------------------------------------------------
 | 【类方法定义】
 |
 |  all_api_methods() from builtins.type
 |         Return a list of all the TradingAlgorithm API methods.
 |
 |  ----------------------------------------------------------------------
 |  【数据说明】
 |
 |  __dict__
 |         dictionary for instance variables (if defined)
 |
 |  __weakref__
 |         list of weak references to the object (if defined)
 |
 |  data_frequency
 |
 |  portfolio
 |
 |  recorded_vars
 |
 |  ----------------------------------------------------------------------
 | 【其他数据和属性定义】
 |
 |  AUTO_INITIALIZE = True
```

__all__

Help on list object:

类定义：list(object)

```
 |  list() -> new empty list
 |  list(iterable) -> new list initialized from iterable's items
 |
 |【类方法定义】
 |
 |  __add__(self, value, /)
 |      Return self+value.
 |
 |  __contains__(self, key, /)
 |      Return key in self.
 |
 |  __delitem__(self, key, /)
 |      Delete self[key].
 |
 |  __eq__(self, value, /)
 |      Return self==value.
 |
 |  __ge__(self, value, /)
 |      Return self>=value.
 |
 |  __getattribute__(self, name, /)
 |      Return getattr(self, name).
 |
 |  __getitem__(...)
 |      x.__getitem__(y) <==> x[y]
 |
 |  __gt__(self, value, /)
 |      Return self>value.
 |
 |  __iadd__(self, value, /)
 |      Implement self+=value.
 |
 |  __imul__(self, value, /)
 |      Implement self*=value.
 |
 |  __init__(self, /, *args, **kwargs)
 |      Initialize self.  See help(type(self)) for accurate signature.
 |
 |  __iter__(self, /)
 |      Implement iter(self).
 |
 |  __le__(self, value, /)
 |      Return self<=value.
 |
 |  __len__(self, /)
 |      Return len(self).
```

```
|
|  __lt__(self, value, /)
|        Return self<value.
|
|  __mul__(self, value, /)
|        Return self*value.n
|
|  __ne__(self, value, /)
|        Return self!=value.
|
|  __new__(*args, **kwargs) from builtins.type
|        Create and return a new object.    See help(type) for accurate signature.
|
|  __repr__(self, /)
|        Return repr(self).
|
|  __reversed__(...)
|        L.__reversed__() -- return a reverse iterator over the list
|
|  __rmul__(self, value, /)
|        Return self*value.
|
|  __setitem__(self, key, value, /)
|        Set self[key] to value.
|
|  __sizeof__(...)
|        L.__sizeof__() -- size of L in memory, in bytes
|
|  append(...)
|        L.append(object) -> None -- append object to end
|
|  clear(...)
|        L.clear() -> None -- remove all items from L
|
|  copy(...)
|        L.copy() -> list -- a shallow copy of L
|
|  count(...)
|        L.count(value) -> integer -- return number of occurrences of value
|
|  extend(...)
|        L.extend(iterable) -> None -- extend list by appending elements from the iterable
|
|  index(...)
|        L.index(value, [start, [stop]]) -> integer -- return first index of value.
|        Raises ValueError if the value is not present.
```

```
|
|  insert(...)
|         L.insert(index, object) -- insert object before index
|
|  pop(...)
|         L.pop([index]) -> item -- remove and return item at index (default last).
|         Raises IndexError if list is empty or index is out of range.
|
|  remove(...)
|         L.remove(value) -> None -- remove first occurrence of value.
|         Raises ValueError if the value is not present.
|
|  reverse(...)
|         L.reverse() -- reverse *IN PLACE*
|
|  sort(...)
|         L.sort(key=None, reverse=False) -> None -- stable sort *IN PLACE*
|
|  ----------------------------------------------------------------------
|【其他数据和属性定义】
|
|  __hash__ = None
```

__builtins__

对象说明：

类定义：dict(object)

```
|  dict() -> new empty dictionary
|  dict(mapping) -> new dictionary initialized from a mapping object's
|         (key, value) pairs
|  dict(iterable) -> new dictionary initialized as if via:
|         d = {}
|         for k, v in iterable:
|                d[k] = v
|  dict(**kwargs) -> new dictionary initialized with the name=value pairs
|         in the keyword argument list.    For example:    dict(one=1, two=2)
|
|【类方法定义】
|
|  __contains__(self, key, /)
|         True if D has a key k, else False.
```

```
 |
 |  __delitem__(self, key, /)
 |      Delete self[key].
 |
 |  __eq__(self, value, /)
 |      Return self==value.
 |
 |  __ge__(self, value, /)
 |      Return self>=value.
 |
 |  __getattribute__(self, name, /)
 |      Return getattr(self, name).
 |
 |  __getitem__(...)
 |      x.__getitem__(y) <==> x[y]
 |
 |  __gt__(self, value, /)
 |      Return self>value.
 |
 |  __init__(self, /, *args, **kwargs)
 |      Initialize self.  See help(type(self)) for accurate signature.
 |
 |  __iter__(self, /)
 |      Implement iter(self).
 |
 |  __le__(self, value, /)
 |      Return self<=value.
 |
 |  __len__(self, /)
 |      Return len(self).
 |
 |  __lt__(self, value, /)
 |      Return self<value.
 |
 |  __ne__(self, value, /)
 |      Return self!=value.
 |
 |  __new__(*args, **kwargs) from builtins.type
 |      Create and return a new object.  See help(type) for accurate signature.
 |
 |  __repr__(self, /)
 |      Return repr(self).
 |
 |  __setitem__(self, key, value, /)
 |      Set self[key] to value.
 |
 |  __sizeof__(...)
 |      D.__sizeof__() -> size of D in memory, in bytes
 |
 |  clear(...)
 |      D.clear() -> None.  Remove all items from D.
 |
 |  copy(...)
 |      D.copy() -> a shallow copy of D
 |
 |  fromkeys(iterable, value=None, /) from builtins.type
 |      Returns a new dict with keys from iterable and values equal to value.
 |
 |  get(...)
 |      D.get(k[,d]) -> D[k] if k in D, else d.  d defaults to None.
 |
 |  items(...)
 |      D.items() -> a set-like object providing a view on D's items
 |
 |  keys(...)
 |      D.keys() -> a set-like object providing a view on D's keys
 |
 |  pop(...)
 |      D.pop(k[,d]) -> v, remove specified key and return the corresponding value.
 |      If key is not found, d is returned if given, otherwise KeyError is raised
 |
 |  popitem(...)
 |      D.popitem() -> (k, v), remove and return some (key, value) pair as a
 |      2-tuple; but raise KeyError if D is empty.
 |
 |  setdefault(...)
 |      D.setdefault(k[,d]) -> D.get(k,d), also set D[k]=d if k not in D
 |
 |  update(...)
 |      D.update([E, ]**F) -> None.  Update D from dict/iterable E and F.
 |      If E is present and has a .keys() method, then does:  for k in E: D[k] = E[k]
 |      If E is present and lacks a .keys() method, then does:  for k, v in E: D[k] = v
 |      In either case, this is followed by: for k in F:  D[k] = F[k]
 |
 |  values(...)
 |      D.values() -> an object providing a view on D's values
 |
 |  ----------------------------------------------------------------------
 |  【其他数据和属性定义】
 |
 |  __hash__ = None
```

__cached__

没有找到相关文档： '\zipline__pycache____init__.cpython-35.pyc'.
请使用 help 命令，查找相关文档。

__doc__

没有找到相关文档： 'Zipline'.
请使用 help 命令，查找相关文档。

__file__

没有找到相关文档： '\zipline__init__.py'.
请使用 help 命令，查找相关文档。

__loader__

```
Help on SourceFileLoader in module importlib._bootstrap_external object:

类定义：SourceFileLoader(FileLoader, SourceLoader)
 |  Concrete implementation of SourceLoader using the file system.
 |
 |  【方法调用顺序】
 |          SourceFileLoader
|        FileLoader
|        SourceLoader
|        _LoaderBasics
|        builtins.object
|
|  【类方法定义】
|
|  path_stats(self, path)
|        Return the metadata for the path.
|
|  set_data(self, path, data, *, _mode=438)
|        Write bytes data to a file.
|
|  ----------------------------------------------------------------------
|  方法源自：FileLoader:
|
|  __eq__(self, other)
|        Return self==value.
|
|  __hash__(self)
|        Return hash(self).
|
|  __init__(self, fullname, path)
|        Cache the module name and the path to the file found by the
|        finder.
|
|  get_data(self, path)
|        Return the data from path as raw bytes.
|
|  get_filename(self, name=None, *args, **kwargs)
|        Return the path to the source file as found by the finder.
|
|  load_module(self, name=None, *args, **kwargs)
|        Load a module from a file.
|
|        This method is deprecated.   Use exec_module() instead.
|
|  ----------------------------------------------------------------------
|  数据说明源自：  FileLoader:
|
|  __dict__
|        dictionary for instance variables (if defined)
|
|  __weakref__
|        list of weak references to the object (if defined)
|
|  ----------------------------------------------------------------------
|  方法源自：SourceLoader:
|
|  get_code(self, fullname)
|      Concrete implementation of InspectLoader.get_code.
|
|      Reading of bytecode requires path_stats to be implemented. To write
|      bytecode, set_data must also be implemented.
|
|  get_source(self, fullname)
|      Concrete implementation of InspectLoader.get_source.
|
|  path_mtime(self, path)
|      Optional method that returns the modification time (an int) for the
|      specified path, where path is a str.
|
|      Raises IOError when the path cannot be handled.
|
|  source_to_code(self, data, path, *, _optimize=-1)
|      Return the code object compiled from source.
|
|      The 'data' argument can be any object type that compile() supports.
|
|  ----------------------------------------------------------------------
|  方法源自：_LoaderBasics:
|
|  create_module(self, spec)
|      Use default semantics for module creation.
|
|  exec_module(self, module)
|      Execute the module.
|
|  is_package(self, fullname)
|      Concrete implementation of InspectLoader.is_package by checking if
|      the path returned by get_filename has a filename of '__init__.py'.
```

__name__

所属模块包：zipline:

【名称】

```
    zipline - Zipline

【模块包内容】
    algorithm
    api
    data (package)
    errors
    finance (package)
    gens (package)
    history (package)
    protocol
    sources (package)
    test_algorithms
    transforms (package)
    utils (package)
    version

【类定义】
    builtins.object
        zipline.algorithm.TradingAlgorithm

    类定义：TradingAlgorithm(builtins.object)
     |  Base class for trading algorithms. Inherit and overload
     |  initialize() and handle_data(data).
     |
     |  A new algorithm could look like this:
     |  ```
     |  from zipline.api import order
     |
     |  def initialize(context):
     |      context.sid = 'AAPL'
     |      context.amount = 100
     |
     |  def handle_data(self, data):
     |      sid = context.sid
     |      amount = context.amount
     |      order(sid, amount)
     |  ```
     |  To then to run this algorithm pass these functions to
     |  TradingAlgorithm:
     |
     |  my_algo = TradingAlgorithm(initialize, handle_data)
     |  stats = my_algo.run(data)
     |
     | 【类方法定义】
     |
```

```
|   __init__(self, *args, **kwargs)
|        Initialize sids and other state variables.
|
|        【参数】
|        :Optional:
|             initialize : function
|                  Function that is called with a single
|                  argument at the begnning of the simulation.
|             handle_data : function
|                  Function that is called with 2 arguments
|                  (context and data) on every bar.
|             script : str
|                  Algoscript that contains initialize and
|                  handle_data function definition.
|             data_frequency : str (daily, hourly or minutely)
|                 The duration of the bars.
|             capital_base : float <default: 1.0e5>
|                 How much capital to start with.
|             instant_fill : bool <default: False>
|                 Whether to fill orders immediately or on next bar.
|
|   __repr__(self)
|        N.B. this does not yet represent a string that can be used
|        to instantiate an exact copy of an algorithm.
|
|        However, it is getting close, and provides some value as something
|        that can be inspected interactively.
|
|   add_history(self, bar_count, frequency, field, ffill=True)
|
|   add_transform(self, transform_class, tag, *args, **kwargs)
|        Add a single-sid, sequential transform to the model.
|
|        【参数】
|             transform_class : class
|                  Which transform to use. E.g. mavg.
|             tag : str
|                  How to name the transform. Can later be access via:
|                  data[sid].tag()
|
|        Extra args and kwargs will be forwarded to the transform
|        instantiation.
|
|   analyze(self, perf)
|
|   cancel_order(self, order_param)
```

```
 |
 |  get_datetime(self)
 |      Returns a copy of the datetime.
 |
 |  get_generator(self)
 |      Override this method to add new logic to the construction
 |      of the generator. Overrides can use the _create_generator
 |      method to get a standard construction generator.
 |
 |  get_open_orders(self, sid=None)
 |
 |  get_order(self, order_id)
 |
 |  handle_data(self, data)
 |
 |  history(self, bar_count, frequency, field, ffill=True)
 |
 |  initialize(self, *args, **kwargs)
 |      Call self._initialize with `self` made available to Zipline API
 |      functions.
 |
 |  on_dt_changed(self, dt)
 |      Callback triggered by the simulation loop whenever the current dt
 |      changes.
 |
 |      Any logic that should happen exactly once at the start of each datetime
 |      group should happen here.
 |
 |  order(self, sid, amount, limit_price=None, stop_price=None, style=None)
 |      Place an order using the specified parameters.
 |
 |  order_percent(self, sid, percent, limit_price=None, stop_price=None, style=None)
 |      Place an order in the specified security corresponding to the given
 |      percent of the current portfolio value.
 |
 |      Note that percent must expressed as a decimal (0.50 means 50\%).
 |
 |  order_target(self, sid, target, limit_price=None, stop_price=None, style=None)
 |      Place an order to adjust a position to a target number of shares. If
 |      the position doesn't already exist, this is equivalent to placing a new
 |      order. If the position does exist, this is equivalent to placing an
 |      order for the difference between the target number of shares and the
 |      current number of shares.
 |
 |  order_target_percent(self, sid, target, limit_price=None, stop_price=None, style=None)
 |      Place an order to adjust a position to a target percent of the
```

```
|       current portfolio value. If the position doesn't already exist, this is
|       equivalent to placing a new order. If the position does exist, this is
|       equivalent to placing an order for the difference between the target
|       percent and the current percent.
|
|       Note that target must expressed as a decimal (0.50 means 50\%).
|
|   order_target_value(self, sid, target, limit_price=None, stop_price=None, style=None)
|       Place an order to adjust a position to a target value. If
|       the position doesn't already exist, this is equivalent to placing a new
|       order. If the position does exist, this is equivalent to placing an
|       order for the difference between the target value and the
|       current value.
|
|   order_value(self, sid, value, limit_price=None, stop_price=None, style=None)
|       Place an order by desired value rather than desired number of shares.
|       If the requested sid is found in the universe, the requested value is
|       divided by its price to imply the number of shares to transact.
|
|       value > 0 :: Buy/Cover
|       value < 0 :: Sell/Short
|       Market order:    order(sid, value)
|       Limit order:     order(sid, value, limit_price)
|       Stop order:      order(sid, value, None, stop_price)
|       StopLimit order: order(sid, value, limit_price, stop_price)
|
|   raw_orders(self)
|       Returns the current open orders from the blotter.
|
|       N.B. this is not a property, so that the function can be passed
|       and called back from within a source.
|
|   raw_positions(self)
|       Returns the current portfolio for the algorithm.
|
|       N.B. this is not done as a property, so that the function can be
|       passed and called from within a source.
|
|   record(self, *args, **kwargs)
|       Track and record local variable (i.e. attributes) each day.
|
|   register_trading_control(self, control)
|       Register a new TradingControl to be checked prior to order calls.
|
|   run(self, source, overwrite_sim_params=True, benchmark_return_source=None)
|       Run the algorithm.
```

```
|
|          【参数】
|               source : can be either:
|                        - pandas.DataFrame
|                        - zipline source
|                        - list of sources
|
|                   If pandas.DataFrame is provided, it must have the
|                   following structure:
|                   * column names must consist of ints representing the
|                     different sids
|                   * index must be DatetimeIndex
|                   * array contents should be price info.
|
|          【返回值】
|               daily_stats : pandas.DataFrame
|                 Daily performance metrics such as returns, alpha etc.
|
|  set_commission(self, commission)
|
|  set_logger(self, logger)
|
|  set_long_only(self)
|      Set a rule specifying that this algorithm cannot take short positions.
|
|  set_max_order_count(self, max_count)
|      Set a limit on the number of orders that can be placed within the given
|      time interval.
|
|  set_max_order_size(self, sid=None, max_shares=None, max_notional=None)
|      Set a limit on the number of shares and/or dollar value of any single
|      order placed for sid.   Limits are treated as absolute values and are
|      enforced at the time that the algo attempts to place an order for sid.
|
|      If an algorithm attempts to place an order that would result in
|      exceeding one of these limits, raise a TradingControlException.
|
|  set_max_position_size(self, sid=None, max_shares=None, max_notional=None)
|      Set a limit on the number of shares and/or dollar value held for the
|      given sid. Limits are treated as absolute values and are enforced at
|      the time that the algo attempts to place an order for sid. This means
|      that it's possible to end up with more than the max number of shares
|      due to splits/dividends, and more than the max notional due to price
|      improvement.
|
|      If an algorithm attempts to place an order that would result in
|          increasing the absolute value of shares/dollar value exceeding one of
|          these limits, raise a TradingControlException.
|
|   set_slippage(self, slippage)
|
|   set_sources(self, sources)
|
|   set_transact(self, transact)
|          Set the method that will be called to create a
|          transaction from open orders and trade events.
|
|   set_transforms(self, transforms)
|
|   update_dividends(self, dividend_frame)
|          Set DataFrame used to process dividends.    DataFrame columns should
|          contain at least the entries in zp.DIVIDEND_FIELDS.
|
|   updated_portfolio(self)
|
|   validate_order_params(self, sid, amount, limit_price, stop_price, style)
|          Helper method for validating parameters to the order API function.
|
|          Raises an UnsupportedOrderParameters if invalid arguments are found.
|
|   ----------------------------------------------------------------------
|  【类方法定义】
|
|   all_api_methods() from builtins.type
|          Return a list of all the TradingAlgorithm API methods.
|
|   ----------------------------------------------------------------------
|   【数据说明】
|
|   __dict__
|          dictionary for instance variables (if defined)
|
|   __weakref__
|          list of weak references to the object (if defined)
|
|   data_frequency
|
|   portfolio
|
|   recorded_vars
|
|   ----------------------------------------------------------------------
```

```
 |  【其他数据和属性定义】
 |
 |  AUTO_INITIALIZE = True
```

【数据】
    __all__ = ['data', 'finance', 'gens', 'utils', 'transforms', 'api', 'T...

VERSION
    0.7.0

【文件】：   \zipline__init__.py

__package__

所属模块包：zipline:

【名称】
    zipline - Zipline

【模块包内容】
    algorithm
    api
    data (package)
    errors
    finance (package)
    gens (package)
    history (package)
    protocol
    sources (package)
    test_algorithms
    transforms (package)
    utils (package)
    version

【类定义】
    builtins.object
        zipline.algorithm.TradingAlgorithm

    类定义：TradingAlgorithm(builtins.object)
     | Base class for trading algorithms. Inherit and overload

```
|  initialize() and handle_data(data).
|
|  A new algorithm could look like this:
|  ```
|  from zipline.api import order
|
|  def initialize(context):
|          context.sid = 'AAPL'
|          context.amount = 100
|
|  def handle_data(self, data):
|          sid = context.sid
|          amount = context.amount
|          order(sid, amount)
|  ```
|  To then to run this algorithm pass these functions to
|  TradingAlgorithm:
|
|  my_algo = TradingAlgorithm(initialize, handle_data)
|  stats = my_algo.run(data)
|
|  【类方法定义】
|
|  __init__(self, *args, **kwargs)
|          Initialize sids and other state variables.
|
|          【参数】
|          :Optional:
|                  initialize : function
|                          Function that is called with a single
|                          argument at the begninning of the simulation.
|                  handle_data : function
|                          Function that is called with 2 arguments
|                          (context and data) on every bar.
|                  script : str
|                          Algoscript that contains initialize and
|                          handle_data function definition.
|                  data_frequency : str (daily, hourly or minutely)
|                        The duration of the bars.
|                  capital_base : float <default: 1.0e5>
|                        How much capital to start with.
|                  instant_fill : bool <default: False>
|                        Whether to fill orders immediately or on next bar.
|
|  __repr__(self)
|          N.B. this does not yet represent a string that can be used
```

```
|          to instantiate an exact copy of an algorithm.
|
|          However, it is getting close, and provides some value as something
|          that can be inspected interactively.
|
|   add_history(self, bar_count, frequency, field, ffill=True)
|
|   add_transform(self, transform_class, tag, *args, **kwargs)
|          Add a single-sid, sequential transform to the model.
|
|          【参数】
|               transform_class : class
|                    Which transform to use. E.g. mavg.
|               tag : str
|                    How to name the transform. Can later be access via:
|                    data[sid].tag()
|
|          Extra args and kwargs will be forwarded to the transform
|          instantiation.
|
|   analyze(self, perf)
|
|   cancel_order(self, order_param)
|
|   get_datetime(self)
|          Returns a copy of the datetime.
|
|   get_generator(self)
|          Override this method to add new logic to the construction
|          of the generator. Overrides can use the _create_generator
|          method to get a standard construction generator.
|
|   get_open_orders(self, sid=None)
|
|   get_order(self, order_id)
|
|   handle_data(self, data)
|
|   history(self, bar_count, frequency, field, ffill=True)
|
|   initialize(self, *args, **kwargs)
|          Call self._initialize with `self` made available to Zipline API
|          functions.
|
|   on_dt_changed(self, dt)
|          Callback triggered by the simulation loop whenever the current dt
```

```
 |      changes.
 |
 |      Any logic that should happen exactly once at the start of each datetime
 |      group should happen here.
 |
 |  order(self, sid, amount, limit_price=None, stop_price=None, style=None)
 |      Place an order using the specified parameters.
 |
 |  order_percent(self, sid, percent, limit_price=None, stop_price=None, style=None)
 |      Place an order in the specified security corresponding to the given
 |      percent of the current portfolio value.
 |
 |      Note that percent must expressed as a decimal (0.50 means 50\%).
 |
 |  order_target(self, sid, target, limit_price=None, stop_price=None, style=None)
 |      Place an order to adjust a position to a target number of shares. If
 |      the position doesn't already exist, this is equivalent to placing a new
 |      order. If the position does exist, this is equivalent to placing an
 |      order for the difference between the target number of shares and the
 |      current number of shares.
 |
 |  order_target_percent(self, sid, target, limit_price=None, stop_price=None, style=None)
 |      Place an order to adjust a position to a target percent of the
 |      current portfolio value. If the position doesn't already exist, this is
 |      equivalent to placing a new order. If the position does exist, this is
 |      equivalent to placing an order for the difference between the target
 |      percent and the current percent.
 |
 |      Note that target must expressed as a decimal (0.50 means 50\%).
 |
 |  order_target_value(self, sid, target, limit_price=None, stop_price=None, style=None)
 |      Place an order to adjust a position to a target value. If
 |      the position doesn't already exist, this is equivalent to placing a new
 |      order. If the position does exist, this is equivalent to placing an
 |      order for the difference between the target value and the
 |      current value.
 |
 |  order_value(self, sid, value, limit_price=None, stop_price=None, style=None)
 |      Place an order by desired value rather than desired number of shares.
 |      If the requested sid is found in the universe, the requested value is
 |      divided by its price to imply the number of shares to transact.
 |
 |      value > 0 :: Buy/Cover
 |      value < 0 :: Sell/Short
 |      Market order:    order(sid, value)
 |      Limit order:     order(sid, value, limit_price)
```

```
|        Stop order:          order(sid, value, None, stop_price)
|        StopLimit order: order(sid, value, limit_price, stop_price)
|
|  raw_orders(self)
|        Returns the current open orders from the blotter.
|
|        N.B. this is not a property, so that the function can be passed
|        and called back from within a source.
|
|  raw_positions(self)
|        Returns the current portfolio for the algorithm.
|
|        N.B. this is not done as a property, so that the function can be
|        passed and called from within a source.
|
|  record(self, *args, **kwargs)
|        Track and record local variable (i.e. attributes) each day.
|
|  register_trading_control(self, control)
|        Register a new TradingControl to be checked prior to order calls.
|
|  run(self, source, overwrite_sim_params=True, benchmark_return_source=None)
|        Run the algorithm.
|
|        【参数】
|            source : can be either:
|                          - pandas.DataFrame
|                          - zipline source
|                          - list of sources
|
|                If pandas.DataFrame is provided, it must have the
|                following structure:
|                * column names must consist of ints representing the
|                  different sids
|                * index must be DatetimeIndex
|                * array contents should be price info.
|
|        【返回值】
|            daily_stats : pandas.DataFrame
|              Daily performance metrics such as returns, alpha etc.
|
|  set_commission(self, commission)
|
|  set_logger(self, logger)
|
|  set_long_only(self)
|         Set a rule specifying that this algorithm cannot take short positions.
|
|   set_max_order_count(self, max_count)
|         Set a limit on the number of orders that can be placed within the given
|         time interval.
|
|   set_max_order_size(self, sid=None, max_shares=None, max_notional=None)
|         Set a limit on the number of shares and/or dollar value of any single
|         order placed for sid.    Limits are treated as absolute values and are
|         enforced at the time that the algo attempts to place an order for sid.
|
|         If an algorithm attempts to place an order that would result in
|         exceeding one of these limits, raise a TradingControlException.
|
|   set_max_position_size(self, sid=None, max_shares=None, max_notional=None)
|         Set a limit on the number of shares and/or dollar value held for the
|         given sid. Limits are treated as absolute values and are enforced at
|         the time that the algo attempts to place an order for sid. This means
|         that it's possible to end up with more than the max number of shares
|         due to splits/dividends, and more than the max notional due to price
|         improvement.
|
|         If an algorithm attempts to place an order that would result in
|         increasing the absolute value of shares/dollar value exceeding one of
|         these limits, raise a TradingControlException.
|
|   set_slippage(self, slippage)
|
|   set_sources(self, sources)
|
|   set_transact(self, transact)
|         Set the method that will be called to create a
|         transaction from open orders and trade events.
|
|   set_transforms(self, transforms)
|
|   update_dividends(self, dividend_frame)
|         Set DataFrame used to process dividends.    DataFrame columns should
|         contain at least the entries in zp.DIVIDEND_FIELDS.
|
|   updated_portfolio(self)
|
|   validate_order_params(self, sid, amount, limit_price, stop_price, style)
|         Helper method for validating parameters to the order API function.
|
|         Raises an UnsupportedOrderParameters if invalid arguments are found.
```

```
 |
 |  ----------------------------------------------------------------------
 | 【类方法定义】
 |
 |  all_api_methods() from builtins.type
 |      Return a list of all the TradingAlgorithm API methods.
 |
 |  ----------------------------------------------------------------------
 |  【数据说明】
 |
 |  __dict__
 |      dictionary for instance variables (if defined)
 |
 |  __weakref__
 |      list of weak references to the object (if defined)
 |
 |  data_frequency
 |
 |  portfolio
 |
 |  recorded_vars
 |
 |  ----------------------------------------------------------------------
 | 【其他数据和属性定义】
 |
 |  AUTO_INITIALIZE = True
```

【数据】
    __all__ = ['data', 'finance', 'gens', 'utils', 'transforms', 'api', 'T...

VERSION
    0.7.0

【文件】：　　\zipline__init__.py

__path__

Help on list object:

类定义：list(object)

```
|  list() -> new empty list
|  list(iterable) -> new list initialized from iterable's items
|
|  【类方法定义】
|
|  __add__(self, value, /)
|        Return self+value.
|
|  __contains__(self, key, /)
|        Return key in self.
|
|  __delitem__(self, key, /)
|        Delete self[key].
|
|  __eq__(self, value, /)
|        Return self==value.
|
|  __ge__(self, value, /)
|        Return self>=value.
|
|  __getattribute__(self, name, /)
|        Return getattr(self, name).
|
|  __getitem__(...)
|        x.__getitem__(y) <==> x[y]
|
|  __gt__(self, value, /)
|        Return self>value.
|
|  __iadd__(self, value, /)
|        Implement self+=value.
|
|  __imul__(self, value, /)
|        Implement self*=value.
|
|  __init__(self, /, *args, **kwargs)
|        Initialize self.   See help(type(self)) for accurate signature.
|
|  __iter__(self, /)
|        Implement iter(self).
|
|  __le__(self, value, /)
|        Return self<=value.
|
|  __len__(self, /)
|        Return len(self).
```

```
|
|  __lt__(self, value, /)
|      Return self<value.
|
|  __mul__(self, value, /)
|      Return self*value.n
|
|  __ne__(self, value, /)
|      Return self!=value.
|
|  __new__(*args, **kwargs) from builtins.type
|      Create and return a new object.  See help(type) for accurate signature.
|
|  __repr__(self, /)
|      Return repr(self).
|
|  __reversed__(...)
|      L.__reversed__() -- return a reverse iterator over the list
|
|  __rmul__(self, value, /)
|      Return self*value.
|
|  __setitem__(self, key, value, /)
|      Set self[key] to value.
|
|  __sizeof__(...)
|      L.__sizeof__() -- size of L in memory, in bytes
|
|  append(...)
|      L.append(object) -> None -- append object to end
|
|  clear(...)
|      L.clear() -> None -- remove all items from L
|
|  copy(...)
|      L.copy() -> list -- a shallow copy of L
|
|  count(...)
|      L.count(value) -> integer -- return number of occurrences of value
|
|  extend(...)
|      L.extend(iterable) -> None -- extend list by appending elements from the iterable
|
|  index(...)
|      L.index(value, [start, [stop]]) -> integer -- return first index of value.
|      Raises ValueError if the value is not present.
```

```
 |
 |  insert(...)
 |      L.insert(index, object) -- insert object before index
 |
 |  pop(...)
 |      L.pop([index]) -> item -- remove and return item at index (default last).
 |      Raises IndexError if list is empty or index is out of range.
 |
 |  remove(...)
 |      L.remove(value) -> None -- remove first occurrence of value.
 |      Raises ValueError if the value is not present.
 |
 |  reverse(...)
 |      L.reverse() -- reverse *IN PLACE*
 |
 |  sort(...)
 |      L.sort(key=None, reverse=False) -> None -- stable sort *IN PLACE*
 |
 |  ----------------------------------------------------------------------
 | 【其他数据和属性定义】
 |
 |  __hash__ = None
```

__spec__

```
Help on ModuleSpec in module importlib._bootstrap object:

类定义：ModuleSpec(builtins.object)
 |  The specification for a module, used for loading.
 |
 |  A module's spec is the source for information about the module.  For
 |  data associated with the module, including source, use the spec's
 |  loader.
 |
 |  `name` is the absolute name of the module.  `loader` is the loader
 |  to use when loading the module.  `parent` is the name of the
 |  package the module is in.  The parent is derived from the name.
 |
 |  `is_package` determines if the module is considered a package or
 |  not.  On modules this is reflected by the `__path__` attribute.
 |
|  `origin` is the specific location used by the loader from which to
|  load the module, if that information is available.   When filename is
|  set, origin will match.
|
|  `has_location` indicates that a spec's "origin" reflects a location.
|  When this is True, `__file__` attribute of the module is set.
|
|  `cached` is the location of the cached bytecode file, if any.   It
|  corresponds to the `__cached__` attribute.
|
|  `submodule_search_locations` is the sequence of path entries to
|  search when importing submodules.   If set, is_package should be
|  True--and False otherwise.
|
|  Packages are simply modules that (may) have submodules.   If a spec
|  has a non-None value in `submodule_search_locations`, the import
|  system will consider modules loaded from the spec as packages.
|
|  Only finders (see importlib.abc.MetaPathFinder and
|  importlib.abc.PathEntryFinder) should modify ModuleSpec instances.
|
| 【类方法定义】
|
|  __eq__(self, other)
|      Return self==value.
|
|  __init__(self, name, loader, *, origin=None, loader_state=None, is_package=None)
|      Initialize self.   See help(type(self)) for accurate signature.
|
|  __repr__(self)
|      Return repr(self).
|
|  ----------------------------------------------------------------------
|  【数据说明】
|
|  __dict__
|      dictionary for instance variables (if defined)
|
|  __weakref__
|      list of weak references to the object (if defined)
|
|  cached
|
|  has_location
|
|  parent
|          The name of the module's parent.
|
|   ----------------------------------------------------------------------
|  【其他数据和属性定义】
|
|   __hash__ = None
```

__version__

没有找到相关文档： '0.7.0'.
请使用 help 命令，查找相关文档。

algorithm

所属模块：zipline.algorithm in zipline:

【名称】
　　zipline.algorithm

【说明】
```
# Copyright 2014 Quantopian, Inc.
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#          http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
```

【类定义】

```
builtins.object
    TradingAlgorithm

类定义：TradingAlgorithm(builtins.object)
 |  Base class for trading algorithms. Inherit and overload
 |  initialize() and handle_data(data).
 |
 |  A new algorithm could look like this:
 |  ```
 |  from zipline.api import order
 |
 |  def initialize(context):
 |      context.sid = 'AAPL'
 |      context.amount = 100
 |
 |  def handle_data(self, data):
 |      sid = context.sid
 |      amount = context.amount
 |      order(sid, amount)
 |  ```
 |  To then to run this algorithm pass these functions to
 |  TradingAlgorithm:
 |
 |  my_algo = TradingAlgorithm(initialize, handle_data)
 |  stats = my_algo.run(data)
 |
 | 【类方法定义】
 |
 |  __init__(self, *args, **kwargs)
 |      Initialize sids and other state variables.
 |
 |     【参数】
 |      :Optional:
 |          initialize : function
 |              Function that is called with a single
 |              argument at the begninning of the simulation.
 |          handle_data : function
 |              Function that is called with 2 arguments
 |              (context and data) on every bar.
 |          script : str
 |              Algoscript that contains initialize and
 |              handle_data function definition.
 |          data_frequency : str (daily, hourly or minutely)
 |             The duration of the bars.
 |          capital_base : float <default: 1.0e5>
 |             How much capital to start with.
```

```
|               instant_fill : bool <default: False>
|                   Whether to fill orders immediately or on next bar.
|
|  __repr__(self)
|        N.B. this does not yet represent a string that can be used
|        to instantiate an exact copy of an algorithm.
|
|        However, it is getting close, and provides some value as something
|        that can be inspected interactively.
|
|  add_history(self, bar_count, frequency, field, ffill=True)
|
|  add_transform(self, transform_class, tag, *args, **kwargs)
|        Add a single-sid, sequential transform to the model.
|
|        【参数】
|               transform_class : class
|                    Which transform to use. E.g. mavg.
|               tag : str
|                    How to name the transform. Can later be access via:
|                    data[sid].tag()
|
|        Extra args and kwargs will be forwarded to the transform
|        instantiation.
|
|  analyze(self, perf)
|
|  cancel_order(self, order_param)
|
|  get_datetime(self)
|        Returns a copy of the datetime.
|
|  get_generator(self)
|        Override this method to add new logic to the construction
|        of the generator. Overrides can use the _create_generator
|        method to get a standard construction generator.
|
|  get_open_orders(self, sid=None)
|
|  get_order(self, order_id)
|
|  handle_data(self, data)
|
|  history(self, bar_count, frequency, field, ffill=True)
|
|  initialize(self, *args, **kwargs)
|      Call self._initialize with `self` made available to Zipline API
|      functions.
|
|  on_dt_changed(self, dt)
|      Callback triggered by the simulation loop whenever the current dt
|      changes.
|
|      Any logic that should happen exactly once at the start of each datetime
|      group should happen here.
|
|  order(self, sid, amount, limit_price=None, stop_price=None, style=None)
|      Place an order using the specified parameters.
|
|  order_percent(self, sid, percent, limit_price=None, stop_price=None, style=None)
|      Place an order in the specified security corresponding to the given
|      percent of the current portfolio value.
|
|      Note that percent must expressed as a decimal (0.50 means 50\%).
|
|  order_target(self, sid, target, limit_price=None, stop_price=None, style=None)
|      Place an order to adjust a position to a target number of shares. If
|      the position doesn't already exist, this is equivalent to placing a new
|      order. If the position does exist, this is equivalent to placing an
|      order for the difference between the target number of shares and the
|      current number of shares.
|
|  order_target_percent(self, sid, target, limit_price=None, stop_price=None, style=None)
|      Place an order to adjust a position to a target percent of the
|      current portfolio value. If the position doesn't already exist, this is
|      equivalent to placing a new order. If the position does exist, this is
|      equivalent to placing an order for the difference between the target
|      percent and the current percent.
|
|      Note that target must expressed as a decimal (0.50 means 50\%).
|
|  order_target_value(self, sid, target, limit_price=None, stop_price=None, style=None)
|      Place an order to adjust a position to a target value. If
|      the position doesn't already exist, this is equivalent to placing a new
|      order. If the position does exist, this is equivalent to placing an
|      order for the difference between the target value and the
|      current value.
|
|  order_value(self, sid, value, limit_price=None, stop_price=None, style=None)
|      Place an order by desired value rather than desired number of shares.
|      If the requested sid is found in the universe, the requested value is
|      divided by its price to imply the number of shares to transact.
```

```
|
|         value > 0 :: Buy/Cover
|         value < 0 :: Sell/Short
|         Market order:       order(sid, value)
|         Limit order:        order(sid, value, limit_price)
|         Stop order:         order(sid, value, None, stop_price)
|         StopLimit order: order(sid, value, limit_price, stop_price)
|
|   raw_orders(self)
|         Returns the current open orders from the blotter.
|
|         N.B. this is not a property, so that the function can be passed
|         and called back from within a source.
|
|   raw_positions(self)
|         Returns the current portfolio for the algorithm.
|
|         N.B. this is not done as a property, so that the function can be
|         passed and called from within a source.
|
|   record(self, *args, **kwargs)
|         Track and record local variable (i.e. attributes) each day.
|
|   register_trading_control(self, control)
|         Register a new TradingControl to be checked prior to order calls.
|
|   run(self, source, overwrite_sim_params=True, benchmark_return_source=None)
|         Run the algorithm.
|
|         【参数】
|              source : can be either:
|                          - pandas.DataFrame
|                          - zipline source
|                          - list of sources
|
|                  If pandas.DataFrame is provided, it must have the
|                  following structure:
|                  * column names must consist of ints representing the
|                     different sids
|                  * index must be DatetimeIndex
|                  * array contents should be price info.
|
|         【返回值】
|              daily_stats : pandas.DataFrame
|                 Daily performance metrics such as returns, alpha etc.
|
 |  set_commission(self, commission)
 |
 |  set_logger(self, logger)
 |
 |  set_long_only(self)
 |      Set a rule specifying that this algorithm cannot take short positions.
 |
 |  set_max_order_count(self, max_count)
 |      Set a limit on the number of orders that can be placed within the given
 |      time interval.
 |
 |  set_max_order_size(self, sid=None, max_shares=None, max_notional=None)
 |      Set a limit on the number of shares and/or dollar value of any single
 |      order placed for sid.  Limits are treated as absolute values and are
 |      enforced at the time that the algo attempts to place an order for sid.
 |
 |      If an algorithm attempts to place an order that would result in
 |      exceeding one of these limits, raise a TradingControlException.
 |
 |  set_max_position_size(self, sid=None, max_shares=None, max_notional=None)
 |      Set a limit on the number of shares and/or dollar value held for the
 |      given sid. Limits are treated as absolute values and are enforced at
 |      the time that the algo attempts to place an order for sid. This means
 |      that it's possible to end up with more than the max number of shares
 |      due to splits/dividends, and more than the max notional due to price
 |      improvement.
 |
 |      If an algorithm attempts to place an order that would result in
 |      increasing the absolute value of shares/dollar value exceeding one of
 |      these limits, raise a TradingControlException.
 |
 |  set_slippage(self, slippage)
 |
 |  set_sources(self, sources)
 |
 |  set_transact(self, transact)
 |      Set the method that will be called to create a
 |      transaction from open orders and trade events.
 |
 |  set_transforms(self, transforms)
 |
 |  update_dividends(self, dividend_frame)
 |      Set DataFrame used to process dividends.  DataFrame columns should
 |      contain at least the entries in zp.DIVIDEND_FIELDS.
 |
 |  updated_portfolio(self)
 |
 |  validate_order_params(self, sid, amount, limit_price, stop_price, style)
 |      Helper method for validating parameters to the order API function.
 |
 |      Raises an UnsupportedOrderParameters if invalid arguments are found.
 |
 |  ----------------------------------------------------------------------
 |  【类方法定义】
 |
 |  all_api_methods() from builtins.type
 |      Return a list of all the TradingAlgorithm API methods.
 |
 |  ----------------------------------------------------------------------
 |  【数据说明】
 |
 |  __dict__
 |      dictionary for instance variables (if defined)
 |
 |  __weakref__
 |      list of weak references to the object (if defined)
 |
 |  data_frequency
 |
 |  portfolio
 |
 |  recorded_vars
 |
 |  ----------------------------------------------------------------------
 |  【其他数据和属性定义】
 |
 |  AUTO_INITIALIZE = True
```

【函数】

```
    exec_ = exec(source, globals=None, locals=None, /)
        Execute the given source in the context of globals and locals.

        The source may be a string representing one or more Python statements
        or a code object as returned by compile().
        The globals must be a dictionary and locals can be any mapping,
        defaulting to the current globals and locals.
        If only globals is given, locals defaults to it.
```

【数据】

```
    DEFAULT_CAPITAL_BASE = 100000.0
```

【文件】：　　\zipline\algorithm.py

api

所属模块：zipline.api in zipline:

【名称】

zipline.api

【说明】

```
# Copyright 2014 Quantopian, Inc.
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#       http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
```

【类定义】

```
builtins.object
    zipline.transforms.batch_transform.BatchTransform
zipline.finance.slippage.SlippageModel(builtins.object)
    zipline.finance.slippage.FixedSlippage
    zipline.finance.slippage.VolumeShareSlippage
```

类定义：FixedSlippage(SlippageModel)

```
 |  【方法调用顺序】
 |      FixedSlippage
 |      SlippageModel
 |      builtins.object
 |
 | 【类方法定义】
 |
 |  __init__(self, spread=0.0)
 |      Use the fixed slippage model, which will just add/subtract
 |          a specified spread spread/2 will be added on buys and subtracted
 |          on sells per share
 |
 |  process_order(self, event, order)
 |
 |  ----------------------------------------------------------------------
 | 【其他数据和属性定义】
 |
 |  __abstractmethods__ = frozenset()
 |
 |  ----------------------------------------------------------------------
 |  方法源自：SlippageModel:
 |
 |  __call__(self, event, current_orders, **kwargs)
 |          Call self as a function.
 |
 |  simulate(self, event, current_orders)
 |
 |  ----------------------------------------------------------------------
 |  数据说明源自：  SlippageModel:
 |
 |  __dict__
 |          dictionary for instance variables (if defined)
 |
 |  __weakref__
 |          list of weak references to the object (if defined)
 |
 |  volume_for_bar

类定义：VolumeShareSlippage(SlippageModel)
 |  【方法调用顺序】
 |          VolumeShareSlippage
 |          SlippageModel
 |          builtins.object
 |
 | 【类方法定义】
 |
 |  __init__(self, volume_limit=0.25, price_impact=0.1)
 |          Initialize self.   See help(type(self)) for accurate signature.
 |
 |  __repr__(self)
 |          Return repr(self).
 |
 |  process_order(self, event, order)
 |
 |  ----------------------------------------------------------------------
 |  【其他数据和属性定义】
 |
 |  __abstractmethods__ = frozenset()
 |
 |  ----------------------------------------------------------------------
 |  方法源自：SlippageModel:
 |
 |  __call__(self, event, current_orders, **kwargs)
 |      Call self as a function.
 |
 |  simulate(self, event, current_orders)
 |
 |  ----------------------------------------------------------------------
 |  数据说明源自： SlippageModel:
 |
 |  __dict__
 |      dictionary for instance variables (if defined)
 |
 |  __weakref__
 |      list of weak references to the object (if defined)
 |
 |  volume_for_bar

batch_transform = class BatchTransform(builtins.object)
 |  Base class for batch transforms with a trailing window of
 |  variable length. As opposed to pure EventWindows that get a stream
 |  of events and are bound to a single SID, this class creates stream
 |  of pandas DataFrames with each colum representing a sid.
 |
 |  There are two ways to create a new batch window:
 |  (i) Inherit from BatchTransform and overload get_value(data).
 |      E.g.:
 |      ```
 |      class MyBatchTransform(BatchTransform):
 |          def get_value(self, data):
 |             # compute difference between the means of sid 0 and sid 1
 |             return data[0].mean() - data[1].mean()
 |      ```
 |
 |  (ii) Use the batch_transform decorator.
 |      E.g.:
 |      ```
 |      @batch_transform
 |      def my_batch_transform(data):
 |          return data[0].mean() - data[1].mean()
 |
```

```
 |      ```
 |
 |  In your algorithm you would then have to instantiate
 |  this in the initialize() method:
 |  ```
 |  self.my_batch_transform = MyBatchTransform()
 |  ```
 |
 |  To then use it, inside of the algorithm handle_data(), call the
 |  handle_data() of the BatchTransform and pass it the current event:
 |  ```
 |  result = self.my_batch_transform(data)
 |  ```
 |
 | 【类方法定义】
 |
 |  __call__(self, f)
 |      Call self as a function.
 |
 |  __init__(self, func=None, refresh_period=0, window_length=None, clean_nans=True, sids=None, fields=None, compute_only_full=True, bars='daily', downsample=False)
 |      Instantiate new batch_transform object.
 |
 |     【参数】
 |          func : python function <optional>
 |              If supplied will be called after each refresh_period
 |              with the data panel and all args and kwargs supplied
 |              to the handle_data() call.
 |          refresh_period : int
 |              Interval to wait between advances in the window.
 |          window_length : int
 |              How many days the trailing window should have.
 |          clean_nans : bool <default=True>
 |              Whether to (forward) fill in nans.
 |          sids : list <optional>
 |              Which sids to include in the moving window.  If not
 |              supplied sids will be extracted from incoming
 |              events.
 |          fields : list <optional>
 |              Which fields to include in the moving window
 |              (e.g. 'price'). If not supplied, fields will be
 |              extracted from incoming events.
 |          compute_only_full : bool <default=True>
 |              Only call the user-defined function once the window is
 |              full. Returns None if window is not full yet.
 |          downsample : bool <default=False>
```

```
 |                      If true, downsample bars to daily bars. Otherwise, do nothing.
 |
 |  get_data(self)
 |        Create a pandas.Panel (i.e. 3d DataFrame) from the
 |        events in the current window.
 |
 |        Returns:
 |        The resulting panel looks like this:
 |        index : field_name (e.g. price)
 |        major axis/rows : dt
 |        minor axis/colums : sid
 |
 |  get_transform_value(self, *args, **kwargs)
 |        Call user-defined batch-transform function passing all
 |        arguments.
 |
 |        Note that this will only call the transform if the datapanel
 |        has actually been updated. Otherwise, the previously, cached
 |        value will be returned.
 |
 |  get_value(self, *args, **kwargs)
 |
 |  handle_data(self, data, *args, **kwargs)
 |        Point of entry. Process an event frame.
 |
 |  ----------------------------------------------------------------------
 |  【数据说明】
 |
 |  __dict__
 |        dictionary for instance variables (if defined)
 |
 |  __weakref__
 |        list of weak references to the object (if defined)

【函数】
    add_history(self, bar_count, frequency, field, ffill=True)
          # Decorator that adds the decorated class method as a callable
          # function (wrapped) to zipline.api

    cancel_order(self, order_param)
          # Decorator that adds the decorated class method as a callable
          # function (wrapped) to zipline.api

    get_datetime(self)
          Returns a copy of the datetime.
```

get_open_orders(self, sid=None)
    # Decorator that adds the decorated class method as a callable
    # function (wrapped) to zipline.api

get_order(self, order_id)
    # Decorator that adds the decorated class method as a callable
    # function (wrapped) to zipline.api

history(self, bar_count, frequency, field, ffill=True)
    # Decorator that adds the decorated class method as a callable
    # function (wrapped) to zipline.api

order(self, sid, amount, limit_price=None, stop_price=None, style=None)
    Place an order using the specified parameters.

order_percent(self, sid, percent, limit_price=None, stop_price=None, style=None)
    Place an order in the specified security corresponding to the given
    percent of the current portfolio value.

    Note that percent must expressed as a decimal (0.50 means 50\%).

order_target(self, sid, target, limit_price=None, stop_price=None, style=None)
    Place an order to adjust a position to a target number of shares. If
    the position doesn't already exist, this is equivalent to placing a new
    order. If the position does exist, this is equivalent to placing an
    order for the difference between the target number of shares and the
    current number of shares.

order_target_percent(self, sid, target, limit_price=None, stop_price=None, style=None)
    Place an order to adjust a position to a target percent of the
    current portfolio value. If the position doesn't already exist, this is
    equivalent to placing a new order. If the position does exist, this is
    equivalent to placing an order for the difference between the target
    percent and the current percent.

    Note that target must expressed as a decimal (0.50 means 50\%).

order_target_value(self, sid, target, limit_price=None, stop_price=None, style=None)
    Place an order to adjust a position to a target value. If
    the position doesn't already exist, this is equivalent to placing a new
    order. If the position does exist, this is equivalent to placing an
    order for the difference between the target value and the
    current value.

order_value(self, sid, value, limit_price=None, stop_price=None, style=None)
    Place an order by desired value rather than desired number of shares.

```
        If the requested sid is found in the universe, the requested value is
        divided by its price to imply the number of shares to transact.

        value > 0 :: Buy/Cover
        value < 0 :: Sell/Short
        Market order:       order(sid, value)
        Limit order:        order(sid, value, limit_price)
        Stop order:         order(sid, value, None, stop_price)
        StopLimit order: order(sid, value, limit_price, stop_price)

    record(self, *args, **kwargs)
        Track and record local variable (i.e. attributes) each day.

    set_commission(self, commission)
        # Decorator that adds the decorated class method as a callable
        # function (wrapped) to zipline.api

    set_long_only(self)
        Set a rule specifying that this algorithm cannot take short positions.

    set_max_order_count(self, max_count)
        Set a limit on the number of orders that can be placed within the given
        time interval.

    set_max_order_size(self, sid=None, max_shares=None, max_notional=None)
        Set a limit on the number of shares and/or dollar value of any single
        order placed for sid.   Limits are treated as absolute values and are
        enforced at the time that the algo attempts to place an order for sid.

        If an algorithm attempts to place an order that would result in
        exceeding one of these limits, raise a TradingControlException.

    set_max_position_size(self, sid=None, max_shares=None, max_notional=None)
        Set a limit on the number of shares and/or dollar value held for the
        given sid. Limits are treated as absolute values and are enforced at
        the time that the algo attempts to place an order for sid. This means
        that it's possible to end up with more than the max number of shares
        due to splits/dividends, and more than the max notional due to price
        improvement.

        If an algorithm attempts to place an order that would result in
        increasing the absolute value of shares/dollar value exceeding one of
        these limits, raise a TradingControlException.

    set_slippage(self, slippage)
        # Decorator that adds the decorated class method as a callable
        # function (wrapped) to zipline.api

    symbol(symbol_str, as_of_date=None)
        Default symbol lookup for any source that directly maps the
        symbol to the identifier (e.g. yahoo finance).

        Keyword argument as_of_date is ignored.
```

【数据】

```
    __all__ = ['symbol', 'slippage', 'commission', 'math_utils', 'batch_tr...
```

【文件】：　　　\zipline\api.py

data

所属模块包：zipline.data in zipline:

【名称】

```
    zipline.data
```

【模块包内容】

```
    benchmarks
    loader
    loader_utils
    treasuries
    treasuries_can
```

【函数】

```
    load_bars_from_yahoo(indexes=None, stocks=None, start=None, end=None, adjusted=True)
        Loads data from Yahoo into a panel with the following
        column names for each indicated security:

            - open
            - high
            - low
            - close
            - volume
            - price

        Note that 'price' is Yahoo's 'Adjusted Close', which removes the
        impact of splits and dividends. If the argument 'adjusted' is True, then
        the open, high, low, and close values are adjusted as well.

        :param indexes: Financial indexes to load.
        :type indexes: dict
        :param stocks: Stock closing prices to load.
        :type stocks: list
        :param start: Retrieve prices from start date on.
        :type start: datetime
        :param end: Retrieve prices until end date.
        :type end: datetime
        :param adjusted: Adjust open/high/low/close for splits and dividends.
            The 'price' field is always adjusted.
        :type adjusted: bool

    load_from_yahoo(indexes=None, stocks=None, start=None, end=None, adjusted=True)
        Loads price data from Yahoo into a dataframe for each of the indicated
        securities.  By default, 'price' is taken from Yahoo's 'Adjusted Close',
        which removes the impact of splits and dividends. If the argument
        'adjusted' is False, then the non-adjusted 'close' field is used instead.

        :param indexes: Financial indexes to load.
        :type indexes: dict
        :param stocks: Stock closing prices to load.
        :type stocks: list
        :param start: Retrieve prices from start date on.
        :type start: datetime
        :param end: Retrieve prices until end date.
        :type end: datetime
        :param adjusted: Adjust the price for splits and dividends.
        :type adjusted: bool
```

【数据】

```
    __all__ = ['loader', 'load_from_yahoo', 'load_bars_from_yahoo']
```

【文件】：　　\zipline\data\__init__.py

errors

所属模块：zipline.errors in zipline:

【名称】

zipline.errors

【说明】

```
# Copyright 2013 Quantopian, Inc.
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#         http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
```

【类定义】

```
builtins.Exception(builtins.BaseException)
    ZiplineError
        OrderDuringInitialize
        OverrideCommissionPostInit
        OverrideSlippagePostInit
        RegisterTradingControlPostInit
        TradingControlViolation
        TransactionVolumeExceedsOrder
        TransactionWithNoAmount
        TransactionWithNoVolume
        TransactionWithWrongDirection
        UnsupportedCommissionModel
        UnsupportedOrderParameters
        UnsupportedSlippageModel
        WrongDataForTransform
```

类定义：OrderDuringInitialize(ZiplineError)

```
 |  Raised if order is called during initialize()
 |
 |  【方法调用顺序】
 |      OrderDuringInitialize
 |      ZiplineError
 |      builtins.Exception
 |      builtins.BaseException
 |      builtins.object
 |
|  【其他数据和属性定义】
|
|  msg = '{msg}'
|
|  ----------------------------------------------------------------------
|  方法源自：ZiplineError:
|
|  __init__(self, *args, **kwargs)
|      Initialize self.  See help(type(self)) for accurate signature.
|
|  __repr__ = __str__(self)
|      Return str(self).
|
|  __str__(self)
|      Return str(self).
|
|  __unicode__ = __str__(self)
|      Return str(self).
|
|  ----------------------------------------------------------------------
|  数据说明源自： ZiplineError:
|
|  __weakref__
|      list of weak references to the object (if defined)
|
|  ----------------------------------------------------------------------
|  方法源自：builtins.Exception:
|
|  __new__(*args, **kwargs) from builtins.type
|      Create and return a new object.  See help(type) for accurate signature.
|
|  ----------------------------------------------------------------------
|  方法源自：builtins.BaseException:
|
|  __delattr__(self, name, /)
|      Implement delattr(self, name).
|
|  __getattribute__(self, name, /)
|      Return getattr(self, name).
|
|  __reduce__(...)
|      helper for pickle
|
|  __setattr__(self, name, value, /)
|      Implement setattr(self, name, value).
|
 |  __setstate__(...)
 |
 |  with_traceback(...)
 |      Exception.with_traceback(tb) --
 |      set self.__traceback__ to tb and return self.
 |
 |  ----------------------------------------------------------------------
 |  数据说明源自：  builtins.BaseException:
 |
 |  __cause__
 |      exception cause
 |
 |  __context__
 |      exception context
 |
 |  __dict__
 |
 |  __suppress_context__
 |
 |  __traceback__
 |
 |  args

类定义：OverrideCommissionPostInit(ZiplineError)
 |  Raised if a users script calls override_commission magic
 |  after the initialize method has returned.
 |
 |  【方法调用顺序】
 |      OverrideCommissionPostInit
 |      ZiplineError
 |      builtins.Exception
 |      builtins.BaseException
 |      builtins.object
 |
 |  【其他数据和属性定义】
 |
 |  msg = 'You attempted to override commission outside of ...ll override_...
 |
 |  ----------------------------------------------------------------------
 |  方法源自：ZiplineError:
 |
 |  __init__(self, *args, **kwargs)
 |      Initialize self.  See help(type(self)) for accurate signature.
 |
 |  __repr__ = __str__(self)
 |      Return str(self).
```

```
|
|   __str__(self)
|       Return str(self).
|
|   __unicode__ = __str__(self)
|       Return str(self).
|
|   ----------------------------------------------------------------------
|   数据说明源自：  ZiplineError:
|
|   __weakref__
|       list of weak references to the object (if defined)
|
|   ----------------------------------------------------------------------
|   方法源自：builtins.Exception:
|
|   __new__(*args, **kwargs) from builtins.type
|       Create and return a new object.  See help(type) for accurate signature.
|
|   ----------------------------------------------------------------------
|   方法源自：builtins.BaseException:
|
|   __delattr__(self, name, /)
|       Implement delattr(self, name).
|
|   __getattribute__(self, name, /)
|       Return getattr(self, name).
|
|   __reduce__(...)
|       helper for pickle
|
|   __setattr__(self, name, value, /)
|       Implement setattr(self, name, value).
|
|   __setstate__(...)
|
|   with_traceback(...)
|       Exception.with_traceback(tb) --
|       set self.__traceback__ to tb and return self.
|
|   ----------------------------------------------------------------------
|   数据说明源自：  builtins.BaseException:
|
|   __cause__
|       exception cause
|
|  __context__
|      exception context
|
|  __dict__
|
|  __suppress_context__
|
|  __traceback__
|
|  args
```

类定义：OverrideSlippagePostInit(ZiplineError)

```
|  Common base class for all non-exit exceptions.
|
|  【方法调用顺序】
|      OverrideSlippagePostInit
|      ZiplineError
|      builtins.Exception
|      builtins.BaseException
|      builtins.object
|
| 【其他数据和属性定义】
|
|  msg = 'You attempted to override slippage outside of `i...call overrid...
|
|  ----------------------------------------------------------------------
|  方法源自：ZiplineError:
|
|  __init__(self, *args, **kwargs)
|      Initialize self.  See help(type(self)) for accurate signature.
|
|  __repr__ = __str__(self)
|      Return str(self).
|
|  __str__(self)
|      Return str(self).
|
|  __unicode__ = __str__(self)
|      Return str(self).
|
|  ----------------------------------------------------------------------
|  数据说明源自： ZiplineError:
|
|  __weakref__
|      list of weak references to the object (if defined)
|
|  ----------------------------------------------------------------------
|  方法源自：builtins.Exception:
|
|  __new__(*args, **kwargs) from builtins.type
|      Create and return a new object.  See help(type) for accurate signature.
|
|  ----------------------------------------------------------------------
|  方法源自：builtins.BaseException:
|
|  __delattr__(self, name, /)
|      Implement delattr(self, name).
|
|  __getattribute__(self, name, /)
|      Return getattr(self, name).
|
|  __reduce__(...)
|      helper for pickle
|
|  __setattr__(self, name, value, /)
|      Implement setattr(self, name, value).
|
|  __setstate__(...)
|
|  with_traceback(...)
|      Exception.with_traceback(tb) --
|      set self.__traceback__ to tb and return self.
|
|  ----------------------------------------------------------------------
|  数据说明源自： builtins.BaseException:
|
|  __cause__
|      exception cause
|
|  __context__
|      exception context
|
|  __dict__
|
|  __suppress_context__
|
|  __traceback__
|
|  args
```

类定义：RegisterTradingControlPostInit(ZiplineError)

```
 |  Common base class for all non-exit exceptions.
```

```
|
|  【方法调用顺序】
|       RegisterTradingControlPostInit
|       ZiplineError
|       builtins.Exception
|       builtins.BaseException
|       builtins.object
|
| 【其他数据和属性定义】
|
|  msg = 'You attempted to set a trading control outside o...ntrols may o...
|
|  ----------------------------------------------------------------------
|  方法源自：ZiplineError:
|
|  __init__(self, *args, **kwargs)
|       Initialize self.   See help(type(self)) for accurate signature.
|
|  __repr__ = __str__(self)
|       Return str(self).
|
|  __str__(self)
|       Return str(self).
|
|  __unicode__ = __str__(self)
|       Return str(self).
|
|  ----------------------------------------------------------------------
|  数据说明源自：  ZiplineError:
|
|  __weakref__
|       list of weak references to the object (if defined)
|
|  ----------------------------------------------------------------------
|  方法源自：builtins.Exception:
|
|  __new__(*args, **kwargs) from builtins.type
|       Create and return a new object.   See help(type) for accurate signature.
|
|  ----------------------------------------------------------------------
|  方法源自：builtins.BaseException:
|
|  __delattr__(self, name, /)
|       Implement delattr(self, name).
|
|  __getattribute__(self, name, /)
|        Return getattr(self, name).
|
|  __reduce__(...)
|        helper for pickle
|
|  __setattr__(self, name, value, /)
|        Implement setattr(self, name, value).
|
|  __setstate__(...)
|
|  with_traceback(...)
|        Exception.with_traceback(tb) --
|        set self.__traceback__ to tb and return self.
|
|  ----------------------------------------------------------------------
|  数据说明源自：  builtins.BaseException:
|
|  __cause__
|        exception cause
|
|  __context__
|        exception context
|
|  __dict__
|
|  __suppress_context__
|
|  __traceback__
|
|  args
```

类定义：TradingControlViolation(ZiplineError)

```
|  Raised if an order would violate a constraint set by a TradingControl.
|
|  【方法调用顺序】
|        TradingControlViolation
|        ZiplineError
|        builtins.Exception
|        builtins.BaseException
|        builtins.object
|
|  【其他数据和属性定义】
|
|  msg = 'Order for {amount} shares of {sid} violates trading constraint ...
|
|  ----------------------------------------------------------------------
|  方法源自：ZiplineError:
|
|  __init__(self, *args, **kwargs)
|      Initialize self.  See help(type(self)) for accurate signature.
|
|  __repr__ = __str__(self)
|      Return str(self).
|
|  __str__(self)
|      Return str(self).
|
|  __unicode__ = __str__(self)
|      Return str(self).
|
|  ----------------------------------------------------------------------
|  数据说明源自： ZiplineError:
|
|  __weakref__
|      list of weak references to the object (if defined)
|
|  ----------------------------------------------------------------------
|  方法源自：builtins.Exception:
|
|  __new__(*args, **kwargs) from builtins.type
|      Create and return a new object.  See help(type) for accurate signature.
|
|  ----------------------------------------------------------------------
|  方法源自：builtins.BaseException:
|
|  __delattr__(self, name, /)
|      Implement delattr(self, name).
|
|  __getattribute__(self, name, /)
|      Return getattr(self, name).
|
|  __reduce__(...)
|      helper for pickle
|
|  __setattr__(self, name, value, /)
|      Implement setattr(self, name, value).
|
|  __setstate__(...)
|
|  with_traceback(...)
|      Exception.with_traceback(tb) --
|      set self.__traceback__ to tb and return self.
```

```
 |
 |  ----------------------------------------------------------------------
 |  数据说明源自： builtins.BaseException:
 |
 |  __cause__
 |      exception cause
 |
 |  __context__
 |      exception context
 |
 |  __dict__
 |
 |  __suppress_context__
 |
 |  __traceback__
 |
 |  args

类定义：TransactionVolumeExceedsOrder(ZiplineError)
 |      Raised if a transact call returns a transaction with a volume greater than
 |  the corresponding order.
 |
 |  【方法调用顺序】
 |      TransactionVolumeExceedsOrder
 |      ZiplineError
 |      builtins.Exception
 |      builtins.BaseException
 |      builtins.object
 |
 | 【其他数据和属性定义】
 |
 |  msg = 'Transaction volume of {txn} exceeds the order volume of {order}...
 |
 |  ----------------------------------------------------------------------
 |  方法源自：ZiplineError:
 |
 |  __init__(self, *args, **kwargs)
 |      Initialize self.  See help(type(self)) for accurate signature.
 |
 |  __repr__ = __str__(self)
 |      Return str(self).
 |
 |  __str__(self)
 |      Return str(self).
 |
 |  __unicode__ = __str__(self)
 |          Return str(self).
 |
 |   ----------------------------------------------------------------------
 |   数据说明源自： ZiplineError:
 |
 |   __weakref__
 |          list of weak references to the object (if defined)
 |
 |   ----------------------------------------------------------------------
 |   方法源自：builtins.Exception:
 |
 |   __new__(*args, **kwargs) from builtins.type
 |          Create and return a new object.  See help(type) for accurate signature.
 |
 |   ----------------------------------------------------------------------
 |   方法源自：builtins.BaseException:
 |
 |   __delattr__(self, name, /)
 |          Implement delattr(self, name).
 |
 |   __getattribute__(self, name, /)
 |          Return getattr(self, name).
 |
 |   __reduce__(...)
 |          helper for pickle
 |
 |   __setattr__(self, name, value, /)
 |          Implement setattr(self, name, value).
 |
 |   __setstate__(...)
 |
 |   with_traceback(...)
 |          Exception.with_traceback(tb) --
 |          set self.__traceback__ to tb and return self.
 |
 |   ----------------------------------------------------------------------
 |   数据说明源自： builtins.BaseException:
 |
 |   __cause__
 |          exception cause
 |
 |   __context__
 |          exception context
 |
 |   __dict__
 |
 |   __suppress_context__
 |
 |   __traceback__
 |
 |   args

 类定义：TransactionWithNoAmount(ZiplineError)
 |   Raised if a transact call returns a transaction with zero amount.
 |
 |   【方法调用顺序】
 |       TransactionWithNoAmount
 |       ZiplineError
 |       builtins.Exception
 |       builtins.BaseException
 |       builtins.object
 |
 |  【其他数据和属性定义】
 |
 |   msg = 'Transaction {txn} has an amount of zero.'
 |
 |   ----------------------------------------------------------------------
 |   方法源自：ZiplineError:
 |
 |   __init__(self, *args, **kwargs)
 |       Initialize self.  See help(type(self)) for accurate signature.
 |
 |   __repr__ = __str__(self)
 |       Return str(self).
 |
 |   __str__(self)
 |       Return str(self).
 |
 |   __unicode__ = __str__(self)
 |       Return str(self).
 |
 |   ----------------------------------------------------------------------
 |   数据说明源自：  ZiplineError:
 |
 |   __weakref__
 |       list of weak references to the object (if defined)
 |
 |   ----------------------------------------------------------------------
 |   方法源自：builtins.Exception:
 |
 |   __new__(*args, **kwargs) from builtins.type
 |       Create and return a new object.  See help(type) for accurate signature.
```

```
 |
 |  ----------------------------------------------------------------------
 |  方法源自：builtins.BaseException:
 |
 |  __delattr__(self, name, /)
 |      Implement delattr(self, name).
 |
 |  __getattribute__(self, name, /)
 |      Return getattr(self, name).
 |
 |  __reduce__(...)
 |      helper for pickle
 |
 |  __setattr__(self, name, value, /)
 |      Implement setattr(self, name, value).
 |
 |  __setstate__(...)
 |
 |  with_traceback(...)
 |      Exception.with_traceback(tb) --
 |      set self.__traceback__ to tb and return self.
 |
 |  ----------------------------------------------------------------------
 |  数据说明源自： builtins.BaseException:
 |
 |  __cause__
 |      exception cause
 |
 |  __context__
 |      exception context
 |
 |  __dict__
 |
 |  __suppress_context__
 |
 |  __traceback__
 |
 |  args

类定义：TransactionWithNoVolume(ZiplineError)
 |  Raised if a transact call returns a transaction with zero volume.
 |
 |  【方法调用顺序】
 |      TransactionWithNoVolume
 |      ZiplineError
 |      builtins.Exception
 |          builtins.BaseException
 |          builtins.object
 |
 |  【其他数据和属性定义】
 |
 |  msg = 'Transaction {txn} has a volume of zero.'
 |
 |  ----------------------------------------------------------------------
 |  方法源自：ZiplineError:
 |
 |  __init__(self, *args, **kwargs)
 |          Initialize self.    See help(type(self)) for accurate signature.
 |
 |  __repr__ = __str__(self)
 |          Return str(self).
 |
 |  __str__(self)
 |          Return str(self).
 |
 |  __unicode__ = __str__(self)
 |          Return str(self).
 |
 |  ----------------------------------------------------------------------
 |  数据说明源自：  ZiplineError:
 |
 |  __weakref__
 |          list of weak references to the object (if defined)
 |
 |  ----------------------------------------------------------------------
 |  方法源自：builtins.Exception:
 |
 |  __new__(*args, **kwargs) from builtins.type
 |          Create and return a new object.    See help(type) for accurate signature.
 |
 |  ----------------------------------------------------------------------
 |  方法源自：builtins.BaseException:
 |
 |  __delattr__(self, name, /)
 |          Implement delattr(self, name).
 |
 |  __getattribute__(self, name, /)
 |          Return getattr(self, name).
 |
 |  __reduce__(...)
 |          helper for pickle
 |
  |  __setattr__(self, name, value, /)
  |      Implement setattr(self, name, value).
  |
  |  __setstate__(...)
  |
  |  with_traceback(...)
  |      Exception.with_traceback(tb) --
  |      set self.__traceback__ to tb and return self.
  |
  |  ----------------------------------------------------------------------
  |  数据说明源自： builtins.BaseException:
  |
  |  __cause__
  |      exception cause
  |
  |  __context__
  |      exception context
  |
  |  __dict__
  |
  |  __suppress_context__
  |
  |  __traceback__
  |
  |  args

 类定义：TransactionWithWrongDirection(ZiplineError)
  |  Raised if a transact call returns a transaction with a direction that
  |  does not match the order.
  |
  |  【方法调用顺序】
  |      TransactionWithWrongDirection
  |      ZiplineError
  |      builtins.Exception
  |      builtins.BaseException
  |      builtins.object
  |
  | 【其他数据和属性定义】
  |
  |  msg = 'Transaction {txn} not in same direction as corresponding order ...
  |
  |  ----------------------------------------------------------------------
  |  方法源自：ZiplineError:
  |
  |  __init__(self, *args, **kwargs)
  |      Initialize self.  See help(type(self)) for accurate signature.
```

```
|
|  __repr__ = __str__(self)
|      Return str(self).
|
|  __str__(self)
|      Return str(self).
|
|  __unicode__ = __str__(self)
|      Return str(self).
|
|  ----------------------------------------------------------------------
|  数据说明源自：  ZiplineError:
|
|  __weakref__
|      list of weak references to the object (if defined)
|
|  ----------------------------------------------------------------------
|  方法源自：builtins.Exception:
|
|  __new__(*args, **kwargs) from builtins.type
|      Create and return a new object.  See help(type) for accurate signature.
|
|  ----------------------------------------------------------------------
|  方法源自：builtins.BaseException:
|
|  __delattr__(self, name, /)
|      Implement delattr(self, name).
|
|  __getattribute__(self, name, /)
|      Return getattr(self, name).
|
|  __reduce__(...)
|      helper for pickle
|
|  __setattr__(self, name, value, /)
|      Implement setattr(self, name, value).
|
|  __setstate__(...)
|
|  with_traceback(...)
|      Exception.with_traceback(tb) --
|      set self.__traceback__ to tb and return self.
|
|  ----------------------------------------------------------------------
|  数据说明源自：  builtins.BaseException:
|
 |  __cause__
 |      exception cause
 |
 |  __context__
 |      exception context
 |
 |  __dict__
 |
 |  __suppress_context__
 |
 |  __traceback__
 |
 |  args
```

类定义：UnsupportedCommissionModel(ZiplineError)

```
 |  Raised if a user script calls the override_commission magic
 |  with a commission object that isn't a PerShare, PerTrade or
 |  PerDollar commission
 |
 |  【方法调用顺序】
 |      UnsupportedCommissionModel
 |      ZiplineError
 |      builtins.Exception
 |      builtins.BaseException
 |      builtins.object
 |
 | 【其他数据和属性定义】
 |
 |  msg = 'You attempted to override commission with an unsupported class....
 |
 |  ----------------------------------------------------------------------
 |  方法源自：ZiplineError:
 |
 |  __init__(self, *args, **kwargs)
 |      Initialize self.  See help(type(self)) for accurate signature.
 |
 |  __repr__ = __str__(self)
 |      Return str(self).
 |
 |  __str__(self)
 |      Return str(self).
 |
 |  __unicode__ = __str__(self)
 |      Return str(self).
 |
 |  ----------------------------------------------------------------------
```

```
|  数据说明源自：  ZiplineError:
|
|  __weakref__
|      list of weak references to the object (if defined)
|
|  ----------------------------------------------------------------------
|  方法源自：builtins.Exception:
|
|  __new__(*args, **kwargs) from builtins.type
|      Create and return a new object.  See help(type) for accurate signature.
|
|  ----------------------------------------------------------------------
|  方法源自：builtins.BaseException:
|
|  __delattr__(self, name, /)
|      Implement delattr(self, name).
|
|  __getattribute__(self, name, /)
|      Return getattr(self, name).
|
|  __reduce__(...)
|      helper for pickle
|
|  __setattr__(self, name, value, /)
|      Implement setattr(self, name, value).
|
|  __setstate__(...)
|
|  with_traceback(...)
|      Exception.with_traceback(tb) --
|      set self.__traceback__ to tb and return self.
|
|  ----------------------------------------------------------------------
|  数据说明源自：  builtins.BaseException:
|
|  __cause__
|      exception cause
|
|  __context__
|      exception context
|
|  __dict__
|
|  __suppress_context__
|
|  __traceback__
```

```
 |
 |  args

类定义：UnsupportedOrderParameters(ZiplineError)
 |  Raised if a set of mutually exclusive parameters are passed to an order
 |  call.
 |
 |  【方法调用顺序】
 |      UnsupportedOrderParameters
 |      ZiplineError
 |      builtins.Exception
 |      builtins.BaseException
 |      builtins.object
 |
 | 【其他数据和属性定义】
 |
 |  msg = '{msg}'
 |
 |  ----------------------------------------------------------------------
 |  方法源自：ZiplineError:
 |
 |  __init__(self, *args, **kwargs)
 |      Initialize self.  See help(type(self)) for accurate signature.
 |
 |  __repr__ = __str__(self)
 |      Return str(self).
 |
 |  __str__(self)
 |      Return str(self).
 |
 |  __unicode__ = __str__(self)
 |      Return str(self).
 |
 |  ----------------------------------------------------------------------
 |  数据说明源自：  ZiplineError:
 |
 |  __weakref__
 |      list of weak references to the object (if defined)
 |
 |  ----------------------------------------------------------------------
 |  方法源自：builtins.Exception:
 |
 |  __new__(*args, **kwargs) from builtins.type
 |      Create and return a new object.  See help(type) for accurate signature.
 |
 |  ----------------------------------------------------------------------
 |  方法源自：builtins.BaseException:
 |
 |  __delattr__(self, name, /)
 |      Implement delattr(self, name).
 |
 |  __getattribute__(self, name, /)
 |      Return getattr(self, name).
 |
 |  __reduce__(...)
 |      helper for pickle
 |
 |  __setattr__(self, name, value, /)
 |      Implement setattr(self, name, value).
 |
 |  __setstate__(...)
 |
 |  with_traceback(...)
 |      Exception.with_traceback(tb) --
 |      set self.__traceback__ to tb and return self.
 |
 |  ----------------------------------------------------------------------
 |  数据说明源自： builtins.BaseException:
 |
 |  __cause__
 |      exception cause
 |
 |  __context__
 |      exception context
 |
 |  __dict__
 |
 |  __suppress_context__
 |
 |  __traceback__
 |
 |  args

类定义：UnsupportedSlippageModel(ZiplineError)
 |  Raised if a user script calls the override_slippage magic
 |  with a slipage object that isn't a VolumeShareSlippage or
 |  FixedSlipapge
 |
 |  【方法调用顺序】
 |      UnsupportedSlippageModel
 |      ZiplineError
 |      builtins.Exception
 |          builtins.BaseException
 |          builtins.object
 |
 | 【其他数据和属性定义】
 |
 | msg = 'You attempted to override slippage with an unsup... Please use ...
 |
 | ----------------------------------------------------------------------
 | 方法源自：ZiplineError:
 |
 | __init__(self, *args, **kwargs)
 |       Initialize self.   See help(type(self)) for accurate signature.
 |
 | __repr__ = __str__(self)
 |       Return str(self).
 |
 | __str__(self)
 |       Return str(self).
 |
 | __unicode__ = __str__(self)
 |       Return str(self).
 |
 | ----------------------------------------------------------------------
 | 数据说明源自：  ZiplineError:
 |
 | __weakref__
 |       list of weak references to the object (if defined)
 |
 | ----------------------------------------------------------------------
 | 方法源自：builtins.Exception:
 |
 | __new__(*args, **kwargs) from builtins.type
 |       Create and return a new object.   See help(type) for accurate signature.
 |
 | ----------------------------------------------------------------------
 | 方法源自：builtins.BaseException:
 |
 | __delattr__(self, name, /)
 |       Implement delattr(self, name).
 |
 | __getattribute__(self, name, /)
 |       Return getattr(self, name).
 |
 | __reduce__(...)
 |       helper for pickle
 |
  |  __setattr__(self, name, value, /)
  |      Implement setattr(self, name, value).
  |
  |  __setstate__(...)
  |
  |  with_traceback(...)
  |      Exception.with_traceback(tb) --
  |      set self.__traceback__ to tb and return self.
  |
  |  ----------------------------------------------------------------------
  |  数据说明源自：  builtins.BaseException:
  |
  |  __cause__
  |      exception cause
  |
  |  __context__
  |      exception context
  |
  |  __dict__
  |
  |  __suppress_context__
  |
  |  __traceback__
  |
  |  args
```

```
类定义：WrongDataForTransform(ZiplineError)
 |  Raised whenever a rolling transform is called on an event that
 |  does not have the necessary properties.
 |
 |  【方法调用顺序】
 |      WrongDataForTransform
 |      ZiplineError
 |      builtins.Exception
 |      builtins.BaseException
 |      builtins.object
 |
 | 【其他数据和属性定义】
 |
 |  msg = '{transform} requires {fields}. Event cannot be processed.'
 |
 |  ----------------------------------------------------------------------
 |  方法源自：ZiplineError:
 |
 |  __init__(self, *args, **kwargs)
 |      Initialize self.  See help(type(self)) for accurate signature.
```

```
|
|  __repr__ = __str__(self)
|        Return str(self).
|
|  __str__(self)
|        Return str(self).
|
|  __unicode__ = __str__(self)
|        Return str(self).
|
|  ----------------------------------------------------------------------
|  数据说明源自： ZiplineError:
|
|  __weakref__
|        list of weak references to the object (if defined)
|
|  ----------------------------------------------------------------------
|  方法源自：builtins.Exception:
|
|  __new__(*args, **kwargs) from builtins.type
|        Create and return a new object.  See help(type) for accurate signature.
|
|  ----------------------------------------------------------------------
|  方法源自：builtins.BaseException:
|
|  __delattr__(self, name, /)
|        Implement delattr(self, name).
|
|  __getattribute__(self, name, /)
|        Return getattr(self, name).
|
|  __reduce__(...)
|        helper for pickle
|
|  __setattr__(self, name, value, /)
|        Implement setattr(self, name, value).
|
|  __setstate__(...)
|
|  with_traceback(...)
|        Exception.with_traceback(tb) --
|        set self.__traceback__ to tb and return self.
|
|  ----------------------------------------------------------------------
|  数据说明源自： builtins.BaseException:
|
 |  __cause__
 |      exception cause
 |
 |  __context__
 |      exception context
 |
 |  __dict__
 |
 |  __suppress_context__
 |
 |  __traceback__
 |
 |  args
```

类定义：ZiplineError(builtins.Exception)

```
 |  Common base class for all non-exit exceptions.
 |
 |  【方法调用顺序】
 |      ZiplineError
 |      builtins.Exception
 |      builtins.BaseException
 |      builtins.object
 |
 | 【类方法定义】
 |
 |  __init__(self, *args, **kwargs)
 |      Initialize self.  See help(type(self)) for accurate signature.
 |
 |  __repr__ = __str__(self)
 |
 |  __str__(self)
 |      Return str(self).
 |
 |  __unicode__ = __str__(self)
 |
 |  ----------------------------------------------------------------------
 |  【数据说明】
 |
 |  __weakref__
 |      list of weak references to the object (if defined)
 |
 |  ----------------------------------------------------------------------
 | 【其他数据和属性定义】
 |
 |  msg = None
 |
 |  ----------------------------------------------------------------------
 |  方法源自：builtins.Exception:
 |
 |  __new__(*args, **kwargs) from builtins.type
 |      Create and return a new object.  See help(type) for accurate signature.
 |
 |  ----------------------------------------------------------------------
 |  方法源自：builtins.BaseException:
 |
 |  __delattr__(self, name, /)
 |      Implement delattr(self, name).
 |
 |  __getattribute__(self, name, /)
 |      Return getattr(self, name).
 |
 |  __reduce__(...)
 |      helper for pickle
 |
 |  __setattr__(self, name, value, /)
 |      Implement setattr(self, name, value).
 |
 |  __setstate__(...)
 |
 |  with_traceback(...)
 |      Exception.with_traceback(tb) --
 |      set self.__traceback__ to tb and return self.
 |
 |  ----------------------------------------------------------------------
 |  数据说明源自： builtins.BaseException:
 |
 |  __cause__
 |      exception cause
 |
 |  __context__
 |      exception context
 |
 |  __dict__
 |
 |  __suppress_context__
 |
 |  __traceback__
 |
 |  args
```

【文件】：　　\zipline\errors.py

finance

所属模块包：zipline.finance in zipline:

【名称】

zipline.finance

【说明】

```
# Copyright 2013 Quantopian, Inc.
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#        http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
```

【模块包内容】

blotter
commission
constants
controls
execution
performance (package)
risk (package)
slippage
trading

【数据】

```
__all__ = ['trading', 'execution']
```

【文件】： \zipline\finance\__init__.py

gens

所属模块包：zipline.gens in zipline:

【名称】
zipline.gens

【模块包内容】
composites
tradesimulation
utils

【文件】： \zipline\gens\__init__.py

history

所属模块包：zipline.history in zipline:

【名称】
zipline.history

【说明】
```
# Copyright 2014 Quantopian, Inc.
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
```

【模块包内容】
    history
    history_container

【类定义】
    builtins.object
        zipline.history.history.HistorySpec

```
类定义：HistorySpec(builtins.object)
 |  Maps to the parameters of the history() call made by the algoscript
 |
 |  An object is used here so that get_history calls are not constantly
 |  parsing the parameters and provides values for caching and indexing into
 |  result frames.
 |
 | 【类方法定义】
 |
 |  __init__(self, bar_count, frequency, field, ffill, daily_at_midnight=False)
 |      Initialize self.  See help(type(self)) for accurate signature.
 |
 |  __repr__(self)
 |      Return repr(self).
 |
 |  ----------------------------------------------------------------------
 | 【类方法定义】
 |
 |  spec_key(bar_count, freq_str, field, ffill) from builtins.type
 |      Used as a hash/key value for the HistorySpec.
 |
 |  ----------------------------------------------------------------------
 |  【数据说明】
 |
 |  __dict__
 |      dictionary for instance variables (if defined)
 |
 |  __weakref__
 |      list of weak references to the object (if defined)
 |
 |  ffill
 |      Wrapper around self._ffill that returns False for fields which are not
 |      forward-fillable.
 |
 |  ----------------------------------------------------------------------
 | 【其他数据和属性定义】
 |
```

| FORWARD_FILLABLE = frozenset({'price'})

【函数】

days_index_at_dt(history_spec, algo_dt)
    Get the index of a frame to be used for a get_history call with daily frequency.

index_at_dt(history_spec, algo_dt)
    Returns index of a frame returned by get_history() with the given history_spec and algo_dt.

    The resulting index `@history_spec.bar_count` bars, increasing in units of `@history_spec.frequency`, terminating at the given @algo_dt.

    Note: The last bar of the returned frame represents an as-of-yet incomplete time window, so the delta between the last and second-to-last bars is usually always less than `@history_spec.frequency` for frequencies greater than 1m.

【数据】

__all__ = ['HistorySpec', 'days_index_at_dt', 'index_at_dt', 'history_...

【文件】：　　\zipline\history\__init__.py

ip

protocol

所属模块：zipline.protocol in zipline:

【名称】

zipline.protocol

【说明】

【类定义】

```
builtins.dict(builtins.object)
        Positions
builtins.object
        BarData
        Event
                Order
        Portfolio
        Position
        SIDData
```

```
类定义：BarData(builtins.object)
 |  Holds the event data for all sids for a given dt.
 |
 |  This is what is passed as `data` to the `handle_data` function.
 |
 |  Note: Many methods are analogues of dictionary because of historical
 |  usage of what this replaced as a dictionary subclass.
 |
 |  【类方法定义】
 |
 |  __contains__(self, name)
 |
 |  __delitem__(self, name)
 |
 |  __getitem__(self, name)
 |
 |  __init__(self, data=None)
 |          Initialize self.   See help(type(self)) for accurate signature.
 |
 |  __iter__(self)
 |
 |  __len__(self)
 |
 |  __repr__(self)
 |      Return repr(self).
 |
 |  __setitem__(self, name, value)
 |
 |  has_key(self, name)
 |      DEPRECATED: __contains__ is preferred, but this method is for
 |      compatibility with existing algorithms.
 |
 |  items(self)
 |
 |  iteritems(self)
 |
 |  iterkeys(self)
 |
 |  itervalues(self)
 |
 |  keys(self)
 |
 |  values(self)
 |
 |  ----------------------------------------------------------------------
 |  【数据说明】
 |
 |  __dict__
 |      dictionary for instance variables (if defined)
 |
 |  __weakref__
 |      list of weak references to the object (if defined)

类定义：Event(builtins.object)
 | 【类方法定义】
 |
 |  __contains__(self, name)
 |
 |  __delitem__(self, name)
 |
 |  __eq__(self, other)
 |      Return self==value.
 |
 |  __getitem__(self, name)
 |
 |  __init__(self, initial_values=None)
 |      Initialize self.  See help(type(self)) for accurate signature.
 |
 |  __repr__(self)
 |      Return repr(self).
 |
 |  __setitem__(self, name, value)
 |
 |  keys(self)
 |
 |  to_series(self, index=None)
 |
 |  ----------------------------------------------------------------------
 |  【数据说明】
 |
 |  __dict__
 |      dictionary for instance variables (if defined)
 |
 |  __weakref__
 |      list of weak references to the object (if defined)
 |
 |  ----------------------------------------------------------------------
 | 【其他数据和属性定义】
 |
 |  __hash__ = None

类定义：Order(Event)
 |  【方法调用顺序】
 |      Order
 |      Event
 |      builtins.object
 |
 |  方法源自：Event:
 |
 |  __contains__(self, name)
 |
 |  __delitem__(self, name)
 |
 |  __eq__(self, other)
 |      Return self==value.
 |
 |  __getitem__(self, name)
 |
 |  __init__(self, initial_values=None)
 |      Initialize self.  See help(type(self)) for accurate signature.
 |
 |  __repr__(self)
 |      Return repr(self).
 |
 |  __setitem__(self, name, value)
 |
 |  keys(self)
 |
 |  to_series(self, index=None)
 |
 |  ----------------------------------------------------------------------
 |  数据说明源自：  Event:
 |
 |  __dict__
 |      dictionary for instance variables (if defined)
 |
 |  __weakref__
 |      list of weak references to the object (if defined)
 |
 |  ----------------------------------------------------------------------
 |  Data and other attributes inherited from Event:
 |
 |  __hash__ = None

类定义：Portfolio(builtins.object)
 | 【类方法定义】
 |
 |  __getitem__(self, key)
 |
 |  __init__(self)
 |      Initialize self.  See help(type(self)) for accurate signature.
 |
 |  __repr__(self)
 |      Return repr(self).
 |
 |  ----------------------------------------------------------------------
 |  【数据说明】
 |
 |  __dict__
 |      dictionary for instance variables (if defined)
 |
 |  __weakref__
 |      list of weak references to the object (if defined)

类定义：Position(builtins.object)
 | 【类方法定义】
 |
 |  __getitem__(self, key)
 |
 |  __init__(self, sid)
 |      Initialize self.  See help(type(self)) for accurate signature.
 |
 |  __repr__(self)
 |      Return repr(self).
 |
 |  ----------------------------------------------------------------------
 |  【数据说明】
 |
 |  __dict__
 |      dictionary for instance variables (if defined)
 |
 |  __weakref__
 |      list of weak references to the object (if defined)

类定义：Positions(builtins.dict)
 |  dict() -> new empty dictionary
 |  dict(mapping) -> new dictionary initialized from a mapping object's
 |      (key, value) pairs
 |  dict(iterable) -> new dictionary initialized as if via:
 |      d = {}
 |      for k, v in iterable:
 |          d[k] = v
 |  dict(**kwargs) -> new dictionary initialized with the name=value pairs
 |      in the keyword argument list.  For example:  dict(one=1, two=2)
 |
 |  【方法调用顺序】
 |      Positions
 |      builtins.dict
 |      builtins.object
 |
 | 【类方法定义】
 |
 |  __missing__(self, key)
 |
 |  ----------------------------------------------------------------------
 |  【数据说明】
 |
 |  __dict__
 |      dictionary for instance variables (if defined)
 |
 |  __weakref__
 |      list of weak references to the object (if defined)
 |
 |  ----------------------------------------------------------------------
 |  方法源自：builtins.dict:
```

```
 |
 |  __contains__(self, key, /)
 |      True if D has a key k, else False.
 |
 |  __delitem__(self, key, /)
 |      Delete self[key].
 |
 |  __eq__(self, value, /)
 |      Return self==value.
 |
 |  __ge__(self, value, /)
 |      Return self>=value.
 |
 |  __getattribute__(self, name, /)
 |      Return getattr(self, name).
 |
 |  __getitem__(...)
 |      x.__getitem__(y) <==> x[y]
 |
 |  __gt__(self, value, /)
 |      Return self>value.
 |
 |  __init__(self, /, *args, **kwargs)
 |      Initialize self.  See help(type(self)) for accurate signature.
 |
 |  __iter__(self, /)
 |      Implement iter(self).
 |
 |  __le__(self, value, /)
 |      Return self<=value.
 |
 |  __len__(self, /)
 |      Return len(self).
 |
 |  __lt__(self, value, /)
 |      Return self<value.
 |
 |  __ne__(self, value, /)
 |      Return self!=value.
 |
 |  __new__(*args, **kwargs) from builtins.type
 |      Create and return a new object.  See help(type) for accurate signature.
 |
 |  __repr__(self, /)
 |      Return repr(self).
 |
 |   __setitem__(self, key, value, /)
 |         Set self[key] to value.
 |
 |   __sizeof__(...)
 |         D.__sizeof__() -> size of D in memory, in bytes
 |
 |   clear(...)
 |         D.clear() -> None.    Remove all items from D.
 |
 |   copy(...)
 |         D.copy() -> a shallow copy of D
 |
 |   fromkeys(iterable, value=None, /) from builtins.type
 |         Returns a new dict with keys from iterable and values equal to value.
 |
 |   get(...)
 |         D.get(k[,d]) -> D[k] if k in D, else d.    d defaults to None.
 |
 |   items(...)
 |         D.items() -> a set-like object providing a view on D's items
 |
 |   keys(...)
 |         D.keys() -> a set-like object providing a view on D's keys
 |
 |   pop(...)
 |         D.pop(k[,d]) -> v, remove specified key and return the corresponding value.
 |         If key is not found, d is returned if given, otherwise KeyError is raised
 |
 |   popitem(...)
 |         D.popitem() -> (k, v), remove and return some (key, value) pair as a
 |         2-tuple; but raise KeyError if D is empty.
 |
 |   setdefault(...)
 |         D.setdefault(k[,d]) -> D.get(k,d), also set D[k]=d if k not in D
 |
 |   update(...)
 |         D.update([E, ]**F) -> None.    Update D from dict/iterable E and F.
 |         If E is present and has a .keys() method, then does:    for k in E: D[k] = E[k]
 |         If E is present and lacks a .keys() method, then does:    for k, v in E: D[k] = v
 |         In either case, this is followed by: for k in F:    D[k] = F[k]
 |
 |   values(...)
 |         D.values() -> an object providing a view on D's values
 |
 |   ----------------------------------------------------------------------
 |   Data and other attributes inherited from builtins.dict:
  |
  |  __hash__ = None

 类定义：SIDData(builtins.object)
  | 【类方法定义】
  |
  |  __contains__(self, name)
  |
  |  __getitem__(self, name)
  |
  |  __init__(self, initial_values=None)
  |      Initialize self.  See help(type(self)) for accurate signature.
  |
  |  __len__(self)
  |
  |  __repr__(self)
  |      Return repr(self).
  |
  |  __setitem__(self, name, value)
  |
  |  ----------------------------------------------------------------------
  |  【数据说明】
  |
  |  __dict__
  |      dictionary for instance variables (if defined)
  |
  |  __weakref__
  |      list of weak references to the object (if defined)
  |
  |  datetime
  |      Provides an alias from data['foo'].datetime -> data['foo'].dt
  |
  |      `datetime` was previously provided by adding a seperate `datetime`
  |      member of the SIDData object via a generator that wrapped the incoming
  |      data feed and added the field to each equity event.
  |
  |      This alias is intended to be temporary, to provide backwards
  |      compatibility with existing algorithms, but should be considered
  |      deprecated, and may be removed in the future.

 【函数】
     dividend_payment(data=None)
         Take a dictionary whose values are in DIVIDEND_PAYMENT_FIELDS and return a
         series representing the payment of a dividend.

         Ids are assigned to each historical dividend in
```

PerformanceTracker.update_dividends. They are guaranteed to be unique
integers with the context of a single simulation. If @data is non-empty, a
id is required to identify the historical dividend associated with this
payment.

Additionally, if @data is non-empty, either data['cash_amount'] should be
nonzero or data['payment_sid'] should be a security identifier and
data['share_count'] should be nonzero.

The returned Series is given its id value as a name so that concatenating
payments results in a DataFrame indexed by id.　(Note, however, that the
name value is not used to construct an index when this series is returned
by function passed to `DataFrame.apply`.　In such a case, pandas preserves
the index of the DataFrame on which `apply` is being called.)

【数据】
DATASOURCE_TYPE = <zipline.utils.protocol_utils.Enum.<locals>.cstruct ...
DIVIDEND_FIELDS = ['declared_date', 'ex_date', 'gross_amount', 'net_am...
DIVIDEND_PAYMENT_FIELDS = ['id', 'payment_sid', 'cash_amount', 'share_...

【文件】：　　\zipline\protocol.py

sources

所属模块包：zipline.sources in zipline:

【名称】
zipline.sources

【模块包内容】
data_frame_source
data_source
simulated
test_source

【类定义】
```
builtins.object
    zipline.sources.test_source.SpecificEquityTrades
zipline.sources.data_source.DataSource(builtins.object)
    zipline.sources.data_frame_source.DataFrameSource
    zipline.sources.data_frame_source.DataPanelSource
    zipline.sources.simulated.RandomWalkSource

类定义：DataFrameSource(zipline.sources.data_source.DataSource)
 |  Yields all events in event_list that match the given sid_filter.
 |  If no event_list is specified, generates an internal stream of events
 |  to filter.   Returns all events if filter is None.
 |
 |  Configuration options:
 |
 |  sids      : list of values representing simulated internal sids
 |  start    : start date
 |  delta    : timedelta between internal events
 |  filter : filter to remove the sids
 |
 |  【方法调用顺序】
 |        DataFrameSource
 |        zipline.sources.data_source.DataSource
 |        builtins.object
 |
 | 【类方法定义】
 |
 |  __init__(self, data, **kwargs)
 |        Initialize self.   See help(type(self)) for accurate signature.
 |
 |  raw_data_gen(self)
 |
 |  ----------------------------------------------------------------------
 |  【数据说明】
 |
 |  instance_hash
 |        A hash that represents the unique args to the source.
 |
 |  mapping
 |        Mappings of the form:
 |        target_key: (mapping_function, source_key)
 |
 |  raw_data
 |        An iterator that yields the raw datasource,
 |        in chronological order of data, one event at a time.
 |
 |  ----------------------------------------------------------------------
 | 【其他数据和属性定义】
 |
 |  __abstractmethods__ = frozenset()
 |
 |  ----------------------------------------------------------------------
 |  方法源自：zipline.sources.data_source.DataSource:
 |
 |  __iter__(self)
 |
 |  __next__(self)
 |
 |  apply_mapping(self, raw_row)
 |      Override this to hand craft conversion of row.
 |
 |  get_hash(self)
 |
 |  next(self)
 |
 |  ----------------------------------------------------------------------
 |  数据说明源自： zipline.sources.data_source.DataSource:
 |
 |  __dict__
 |      dictionary for instance variables (if defined)
 |
 |  __weakref__
 |      list of weak references to the object (if defined)
 |
 |  event_type
 |
 |  mapped_data

类定义：DataPanelSource(zipline.sources.data_source.DataSource)
 |  Yields all events in event_list that match the given sid_filter.
 |  If no event_list is specified, generates an internal stream of events
 |  to filter.  Returns all events if filter is None.
 |
 |  Configuration options:
 |
 |  sids     : list of values representing simulated internal sids
 |  start    : start date
 |  delta    : timedelta between internal events
 |  filter : filter to remove the sids
 |
 |  【方法调用顺序】
 |      DataPanelSource
 |      zipline.sources.data_source.DataSource
 |      builtins.object
 |
 | 【类方法定义】
 |
|  __init__(self, data, **kwargs)
|       Initialize self.   See help(type(self)) for accurate signature.
|
|  raw_data_gen(self)
|
|  ----------------------------------------------------------------------
|  【数据说明】
|
|  instance_hash
|       A hash that represents the unique args to the source.
|
|  mapping
|       Mappings of the form:
|       target_key: (mapping_function, source_key)
|
|  raw_data
|       An iterator that yields the raw datasource,
|       in chronological order of data, one event at a time.
|
|  ----------------------------------------------------------------------
| 【其他数据和属性定义】
|
|  __abstractmethods__ = frozenset()
|
|  ----------------------------------------------------------------------
|  方法源自：zipline.sources.data_source.DataSource:
|
|  __iter__(self)
|
|  __next__(self)
|
|  apply_mapping(self, raw_row)
|       Override this to hand craft conversion of row.
|
|  get_hash(self)
|
|  next(self)
|
|  ----------------------------------------------------------------------
|  数据说明源自：   zipline.sources.data_source.DataSource:
|
|  __dict__
|       dictionary for instance variables (if defined)
|
|  __weakref__
|       list of weak references to the object (if defined)
```

```
 |
 |  event_type
 |
 |  mapped_data

类定义：RandomWalkSource(zipline.sources.data_source.DataSource)
 |  RandomWalkSource that emits events with prices that follow a
 |  random walk. Will generate valid datetimes that match market hours
 |  of the supplied calendar and can generate emit events with
 |  user-defined frequencies (e.g. minutely).
 |
 |  【方法调用顺序】
 |      RandomWalkSource
 |      zipline.sources.data_source.DataSource
 |      builtins.object
 |
 | 【类方法定义】
 |
 |      __init__(self, start_prices=None, freq='minute', start=None, end=None, calendar=<module 'zipline.utils.tradingcalendar' from '\zipline\utils\tradingcalendar.py'>)
 |      【参数】
 |          start_prices : dict
 |              sid -> starting price.
 |              Default: {0: 100, 1: 500}
 |          freq : str <default='minute'>
 |              Emits events according to freq.
 |              Can be 'day' or 'minute'
 |          start : datetime <default=start of calendar>
 |              Start dt to emit events.
 |          end : datetime <default=end of calendar>
 |              End dt until to which emit events.
 |          calendar : calendar object <default: NYSE>
 |              Calendar to use.
 |              See zipline.utils for different choices.
 |
 |      :Example:
 |          # Assumes you have instantiated your Algorithm
 |          # as myalgo.
 |          myalgo = MyAlgo()
 |          source = RandomWalkSource()
 |          myalgo.run(source)
 |
 |  raw_data_gen(self)
 |
 |  ----------------------------------------------------------------------
 |  【数据说明】
```

```
|
|  instance_hash
|        A hash that represents the unique args to the source.
|
|  mapping
|        Mappings of the form:
|        target_key: (mapping_function, source_key)
|
|  raw_data
|        An iterator that yields the raw datasource,
|        in chronological order of data, one event at a time.
|
|  ----------------------------------------------------------------------
|【其他数据和属性定义】
|
|  __abstractmethods__ = frozenset()
|
|  ----------------------------------------------------------------------
|  方法源自：zipline.sources.data_source.DataSource:
|
|  __iter__(self)
|
|  __next__(self)
|
|  apply_mapping(self, raw_row)
|        Override this to hand craft conversion of row.
|
|  get_hash(self)
|
|  next(self)
|
|  ----------------------------------------------------------------------
|  数据说明源自：  zipline.sources.data_source.DataSource:
|
|  __dict__
|        dictionary for instance variables (if defined)
|
|  __weakref__
|        list of weak references to the object (if defined)
|
|  event_type
|
|  mapped_data

类定义：SpecificEquityTrades(builtins.object)
 |  Yields all events in event_list that match the given sid_filter.
 |  If no event_list is specified, generates an internal stream of events
 |  to filter.   Returns all events if filter is None.
 |
 |  Configuration options:
 |
 |  count    : integer representing number of trades
 |  sids      : list of values representing simulated internal sids
 |  start    : start date
 |  delta    : timedelta between internal events
 |  filter : filter to remove the sids
 |
 |  【类方法定义】
 |
 |  __init__(self, *args, **kwargs)
 |         Initialize self.   See help(type(self)) for accurate signature.
 |
 |  __iter__(self)
 |
 |  __next__(self)
 |
 |  create_fresh_generator(self)
 |
 |  get_hash(self)
 |
 |  next(self)
 |
 |  rewind(self)
 |
 |  update_source_id(self, gen)
 |
 |  ----------------------------------------------------------------------
 |   【数据说明】
 |
 |  __dict__
 |         dictionary for instance variables (if defined)
 |
 |  __weakref__
 |         list of weak references to the object (if defined)
```

【数据】

```
    __all__ = ['DataFrameSource', 'DataPanelSource', 'SpecificEquityTrades...
```

【文件】：   \zipline\sources\__init__.py

transforms

所属模块包：zipline.transforms in zipline:

【名称】

zipline.transforms

【说明】

```
# Copyright 2012 Quantopian, Inc.
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#      http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
```

【模块包内容】

batch_transform
mavg
returns
stddev
ta
utils
vwap

【类定义】

```
builtins.object
    zipline.transforms.batch_transform.BatchTransform
    zipline.transforms.mavg.MovingAverage
    zipline.transforms.returns.Returns
    zipline.transforms.stddev.MovingStandardDev
    zipline.transforms.vwap.MovingVWAP
```

类定义：BatchTransform(builtins.object)

```
 |  Base class for batch transforms with a trailing window of
 |  variable length. As opposed to pure EventWindows that get a stream
 |  of events and are bound to a single SID, this class creates stream
 |  of pandas DataFrames with each colum representing a sid.
 |
 |  There are two ways to create a new batch window:
 |  (i) Inherit from BatchTransform and overload get_value(data).
 |      E.g.:
 |      ```
 |      class MyBatchTransform(BatchTransform):
 |          def get_value(self, data):
 |             # compute difference between the means of sid 0 and sid 1
 |             return data[0].mean() - data[1].mean()
 |      ```
 |
 |  (ii) Use the batch_transform decorator.
 |      E.g.:
 |      ```
 |      @batch_transform
 |      def my_batch_transform(data):
 |          return data[0].mean() - data[1].mean()
 |
 |      ```
 |
 |  In your algorithm you would then have to instantiate
 |  this in the initialize() method:
 |  ```
 |  self.my_batch_transform = MyBatchTransform()
 |  ```
 |
 |  To then use it, inside of the algorithm handle_data(), call the
 |  handle_data() of the BatchTransform and pass it the current event:
 |  ```
 |  result = self.my_batch_transform(data)
 |  ```
 |
 |  【类方法定义】
 |
 |  __call__(self, f)
 |      Call self as a function.
 |
 |   __init__(self, func=None, refresh_period=0, window_length=None, clean_nans=True, sids=None,
fields=None, compute_only_full=True, bars='daily', downsample=False)
 |      Instantiate new batch_transform object.
 |
 |      【参数】
 |          func : python function <optional>
 |              If supplied will be called after each refresh_period
```

```
 |                  with the data panel and all args and kwargs supplied
 |                  to the handle_data() call.
 |              refresh_period : int
 |                  Interval to wait between advances in the window.
 |              window_length : int
 |                  How many days the trailing window should have.
 |              clean_nans : bool <default=True>
 |                  Whether to (forward) fill in nans.
 |              sids : list <optional>
 |                  Which sids to include in the moving window.    If not
 |                  supplied sids will be extracted from incoming
 |                  events.
 |              fields : list <optional>
 |                  Which fields to include in the moving window
 |                  (e.g. 'price'). If not supplied, fields will be
 |                  extracted from incoming events.
 |              compute_only_full : bool <default=True>
 |                  Only call the user-defined function once the window is
 |                  full. Returns None if window is not full yet.
 |              downsample : bool <default=False>
 |                  If true, downsample bars to daily bars. Otherwise, do nothing.
 |
 |  get_data(self)
 |      Create a pandas.Panel (i.e. 3d DataFrame) from the
 |      events in the current window.
 |
 |      Returns:
 |      The resulting panel looks like this:
 |      index : field_name (e.g. price)
 |      major axis/rows : dt
 |      minor axis/colums : sid
 |
 |  get_transform_value(self, *args, **kwargs)
 |      Call user-defined batch-transform function passing all
 |      arguments.
 |
 |      Note that this will only call the transform if the datapanel
 |      has actually been updated. Otherwise, the previously, cached
 |      value will be returned.
 |
 |  get_value(self, *args, **kwargs)
 |
 |  handle_data(self, data, *args, **kwargs)
 |      Point of entry. Process an event frame.
 |
 |  ----------------------------------------------------------------------
 |  【数据说明】
 |
 |  __dict__
 |      dictionary for instance variables (if defined)
 |
 |  __weakref__
 |      list of weak references to the object (if defined)

类定义：MovingAverage(builtins.object)
 |  Class that maintains a dictionary from sids to
 |  MovingAverageEventWindows.  For each sid, we maintain moving
 |  averages over any number of distinct fields (For example, we can
 |  maintain a sid's average volume as well as its average price.)
 |
 | 【类方法定义】
 |
 |  __init__(self, fields='price', market_aware=True, window_length=None, delta=None)
 |      Initialize self.  See help(type(self)) for accurate signature.
 |
 |  create_window(self)
 |      Factory method for self.sid_windows.
 |
 |  update(self, event)
 |      Update the event window for this event's sid.  Return a dict
 |      from tracked fields to moving averages.
 |
 |  ----------------------------------------------------------------------
 |  【数据说明】
 |
 |  __dict__
 |      dictionary for instance variables (if defined)
 |
 |  __weakref__
 |      list of weak references to the object (if defined)

类定义：MovingStandardDev(builtins.object)
 |  Class that maintains a dictionary from sids to
 |  MovingStandardDevWindows.  For each sid, we maintain a the
 |  standard deviation of all events falling within the specified
 |  window.
 |
 | 【类方法定义】
 |
 |  __init__(self, market_aware=True, window_length=None, delta=None)
 |      Initialize self.  See help(type(self)) for accurate signature.
 |
 |  create_window(self)
 |      Factory method for self.sid_windows.
 |
 |  update(self, event)
 |      Update the event window for this event's sid.   Return a dict
 |      from tracked fields to moving averages.
 |
 |  ----------------------------------------------------------------------
 |  【数据说明】
 |
 |  __dict__
 |      dictionary for instance variables (if defined)
 |
 |  __weakref__
 |      list of weak references to the object (if defined)

类定义：MovingVWAP(builtins.object)
 |  Class that maintains a dictionary from sids to VWAPEventWindows.
 |
 | 【类方法定义】
 |
 |  __init__(self, market_aware=True, delta=None, window_length=None)
 |      Initialize self.   See help(type(self)) for accurate signature.
 |
 |  create_window(self)
 |      Factory method for self.sid_windows.
 |
 |  update(self, event)
 |      Update the event window for this event's sid. Returns the
 |      current vwap for the sid.
 |
 |  ----------------------------------------------------------------------
 |  【数据说明】
 |
 |  __dict__
 |      dictionary for instance variables (if defined)
 |
 |  __weakref__
 |      list of weak references to the object (if defined)

类定义：Returns(builtins.object)
 |  Class that maintains a dictionary from sids to the sid's
 |  closing price N trading days ago.
 |
 | 【类方法定义】
 |
|  __init__(self, window_length)
|        Initialize self.   See help(type(self)) for accurate signature.
|
|  update(self, event)
|        Update and return the calculated returns for this event's sid.
|
|  ----------------------------------------------------------------------
|  【数据说明】
|
|  __dict__
|        dictionary for instance variables (if defined)
|
|  __weakref__
|        list of weak references to the object (if defined)
```

【函数】
```
    batch_transform(func)
        Decorator function to use instead of inheriting from BatchTransform.
        For an example on how to use this, see the doc string of BatchTransform.
```

【数据】
```
    __all__ = ['MovingAverage', 'MovingStandardDev', 'MovingVWAP', 'Return...
```

【文件】：   \zipline\transforms\__init__.py

utils

所属模块包：zipline.utils in zipline:

【名称】

zipline.utils

【说明】
```
    # Copyright 2014 Quantopian, Inc.
    #
    # Licensed under the Apache License, Version 2.0 (the "License");
    # you may not use this file except in compliance with the License.
    # You may obtain a copy of the License at
    #
    #      http://www.apache.org/licenses/LICENSE-2.0
```

【模块包内容】

api_support
cli
data
data_source_tables_gen
factory
math_utils
protocol_utils
simfactory
test_utils
tradingcalendar
tradingcalendar_bmf
tradingcalendar_lse
tradingcalendar_tse

【函数】

```
parse_args(argv, ipython_mode=False)
    Parse list of arguments.

    If a config file is provided (via -c), it will read in the
    supplied options and overwrite any global defaults.

    All other directly supplied arguments will overwrite the config
    file settings.

    Arguments:
        * argv : list of strings
            List of arguments, e.g. ['-c', 'my.conf']
        * ipython_mode : bool <default=True>
            Whether to parse IPython specific arguments
            like --local_namespace

    Notes:
    Default settings can be found in zipline.utils.cli.DEFAULTS.

parse_cell_magic(line, cell)
    Parse IPython magic

run_pipeline(print_algo=True, **kwargs)
```

Runs a full zipline pipeline given configuration keyword arguments.

1. Load data (start and end dates can be provided a strings as well as the source and symobls).

2. Instantiate algorithm (supply either algo_text or algofile kwargs containing initialize() and handle_data() functions). If algofile is supplied, will try to look for algofile_analyze.py and append it.

3. Run algorithm (supply capital_base as float).

4. Return performance dataframe.

【参数】

* print_algo : bool <default=True>

  Whether to print the algorithm to command line. Will use pygments syntax coloring if pygments is found.

【数据】

__all__ = ['run_pipeline', 'parse_args', 'parse_cell_magic']

【文件】：　　\zipline\utils\__init__.py