**简单版(对用户来说)策略的详细(hai kan bu dong jiu kan si ni)说明**

对于用户来说，一个策略对应多个 IF x THEN y的组合。比如说，IF 连续五天跌 THEN 卖出100股sh600036股票（x=连续5天跌，y=卖出100股sh600036股票)；

又如，IF macd线由下向上突破0 THEN 买入 100股sh600036；

每个IF x THEN y称作一个Recipe，所以，一个策略(Strategy)等于多个Recipe的组合

比如，某用户看好了sh600036，想在连续五天跌的时候卖出去，由想在macd线由下向上突破0时候买入，那么他制定的策略 = [Recipe1，Recipe2}

IF x THEN y 中，x 是条件(Condition)，y是动作(Action)，显然应该要能够组合不同的条件和动作来组合出不同的Recipe，比如从上面两个Recipe的动作中，可以组合出 IF 连续五天跌 THEN 买入100股sh600036股票。

到这里，用户可以做到的是，新建策略，新建Recipe（选择条件，选择动作），新建Recipe重复若干次，编译策略（转成python代码），保存策略。

但是具体怎么编译？显然只能人工写，比如说，一个条件，叫短期滑动均线低于长期，写成python代码就是：

def handle\_data(context, data):

short\_avg = data.history(symbol(“sh600036”), 'close', 9, '1d').mean() # 9天收盘价平均

long\_avg = data.history(symbol(“sh600036”), 'close', 16, '1d').mean() # 16天收盘价平均

if short\_avg < long\_avg :

# take action

在用户指定了动作之后，我们的工作就是把动作的代码放到 # take action的位置

由于条件和动作都是可以不断添加的，所以我们要定义一些格式，从而可以让我们按照这个格式来添加条件和动作。那么我们的格式至少应该包括，条件/动作的名字，条件/动作的代码模板。

只考虑这两个的话，把上面那个“短期滑动均线低于长期”条件翻译成格式，就是

&&name # &&开头表示一个区域标识，这里是名字, #后面是注释

mva\_compare # 一个区域标识开始，直到文件结束或另一个区域标识之间是区域定义，name区域只识别第一行（\n结束），且开头不为空，只含a-zA-Z0-9以及下划线，空格或换行表示结束

&&compute # 计算条件表达式中要用到的变量的值,请正确填写python代码（以后出现在handle\_data中）

short\_avg = data.history(symbol(“sh600036”), 'close', 9, '1d').mean()

long\_avg = data.history(symbol(“sh600036”), 'close', 16, '1d').mean()

&&condition # 条件表达式

short\_avg < long\_avg

到这里，用户可以做到，自定义新的条件，自定义新的动作，因为定义这些的时候都是要通过写东西完成的，所以界面层提供一个文本框就好了

但是只到这里的话，好像有点弱，因为写的太死了，比如动作那里只能固定买100股，万一我想买1000股，又要重写一个动作，显然比较坑。要搞定这个问题，就要引入变量。拿“短期滑动均线低于长期”这条件作为例子，假如我要的不是9天和16天，是7天和17天，要怎么做呢?在模板那里加点东西就好了

&&name # &&开头表示一个区域标识，这里是名字, #后面是注释

mva\_compare # 一个区域标识开始，直到文件结束或另一个区域标识之间是区域定义，name区域只识别第一行，且开头不为空，只含a-zA-Z0-9以及下划线，空格或换行表示结束

&&description # 描述

to define a condition where short\_mavg < long\_mavg # 可以任意字符任意行

&&variable # 要被替换的变量

short\_term=9, long\_term=16 # 变量之间用逗号隔开，可以出现大小写数字下划线，开头不能是数字；变量名不能是stock（已默认使用）;变量名后加=号表示默认值

&&compute # 计算条件表达式中要用到的变量的值,请正确填写python代码（以后出现在handle\_data中）

short\_avg = data.history(symbol($stock), 'close', $short\_term, '1d').mean()

long\_avg = data.history(symbol($stock), 'close', $long\_term, '1d').mean()

&&condition # 条件表达式

short\_avg < long\_avg

上面这样，就能够定义一次条件，但是适配到不同的策略里面去了，有人要9和16天，有人要7和17天，只要在选完条件之后多选一下就好了，没有默认值的变量一定要让用户填一下，有默认值的假如用户不改，那就是默认值。而且有了变量，我们就可以指定要作用到的股票了。

到这里，用户可以做到，新建策略（选择一支或一系列股票），新建Recipe（选择条件（并指定变量），选择动作（并指定变量）），新建Recipe重复若干次，编译策略（转成python代码），保存策略。

既然可以定义变量了，干脆也能定义常量吧。再加一点东西。

&&name # &&开头表示一个区域标识，这里是名字, #后面是注释

mva\_compare # 一个区域标识开始，直到文件结束或另一个区域标识之间是区域定义，name区域只识别第一行，且开头不为空，只含a-zA-Z0-9以及下划线，空格或换行表示结束

&&description # 描述

to define a condition where short\_mavg < long\_mavg # 可以任意字符任意行

&&variable # 要被替换的变量

max\_share, short\_term=9, long\_term=16 # 变量之间用逗号隔开，可以出现大小写数字下划线，开头不能是数字；变量名不能是stock（默认使用）；变量名后加=号表示默认值

&&const # 常量定义区域，请正确填写python代码（以后出现在initialize中）

from zipline.api import order, symbol

context.stock1 = symbol($stock) # $stock表示应用该策略的股票, 生成代码时候会替换成相应的股票代码，相当于stock是默认的变量

context.max\_shares = $max\_share # $跟着变量名，表示要替换成实际输入

def buy\_buy\_buy():

order(context.stock1, context.max\_shares)

&&compute # 计算条件表达式中要用到的变量的值,请正确填写python代码（以后出现在handle\_data中）

short\_avg = data.history(context.stock1, 'close', $short\_term, '1d').mean()

long\_avg = data.history(context.stock1, 'close', $long\_term, '1d').mean()

&&condition # 条件表达式

short\_avg < long\_avg

到这里，用户可以在定义模板的时候多写点东西，不过对于界面层来说，还是只提供一个文本框就够了。

最后，应该还要支持条件的and,or,not，以及动作的拼接（动作1接动作2），在动作里嵌套Recipe，这步不需要改模板格式。

到这里，用户可以做到，新建策略（选择一支或一系列股票），新建Recipe（选择条件（并指定变量），选择动作（并指定变量）），新建Recipe重复若干次，编译策略（转成python代码），保存策略。

选择条件时候，可以点and/or，这时候，可以选两个条件，两个条件里面还可以选and/or；也可以选not，再点选别的

选择动作时候，可以不断添加新动作，也可以不添加动作，而是添加Recipe