



# 策略代码跨平台迁移指南

## 1. 策略代码迁移的可行性与难点

**执行环境差异：**聚宽 (JoinQuant) 和韬睿量化等研究平台通常运行在云端模拟环境，而 PTrade 与 QMT 属于券商实盘平台，其中 **PTrade** 是云端托管（策略在服务器运行），**QMT** 则需要本地运行（策略在本地电脑/云主机上执行）<sup>1</sup> <sup>2</sup>。这意味着聚宽的策略代码不能直接在目标平台上运行，必须考虑环境差异：在 QMT 上需要先下载行情和财务数据，本地保存后供策略使用<sup>3</sup>；而 PTrade 因与聚宽同源，数据接口和运行机制更接近，可“无缝切换”，但也需注意API细微差异<sup>1</sup>。

**框架与调度差异：**聚宽采用 `initialize(context)` 初始化、`handle_data(context, data)` 主循环，以及 `run_daily()/run_weekly()` 等定时调度函数。PTrade 的代码框架与之类似：也有 `initialize`、`handle_data`，以及可选的 `before_trading_start` / `after_trading_end`，支持通过 `run_interval()` 等设定盘中定时任务<sup>4</sup>。**QMT** 则有自己的一套函数：使用 `init(C)` 初始化、`handlebar(C)` 作为主要回调，每个Bar触发一次（类似于`handle_data`），不区分日频或分钟频的函数<sup>5</sup>。例如，在聚宽中如果用了 `run_daily(func, time='every_bar')` 每Bar执行，迁移到 QMT 需要利用其 `handlebar` 每Bar触发的机制；若聚宽策略按周调仓（`run_weekly`），则在QMT需要自行控制调用频率或判断日期<sup>6</sup>。总之，**QMT**没有聚宽那样的多粒度调度函数，需要通过单一的 `handlebar` 配合时间判断实现等价逻辑<sup>6</sup>。

**API函数差异：**最大的工作量在于 **交易和数据接口的替换**<sup>7</sup>。聚宽API如下单函数 `order()` / `order_target()`、获取历史行情的 `attribute_history()` / `get_price()`、记录日志 `log.info()`、记录指标 `record()` 等，在PTrade/QMT上名称和用法各异。比如：

• **交易下单：**聚宽使用简单的 `order(security, amount)` 或 `order_value(security, value)` 等；PTrade 提供了类似的交易函数（如 `order_target()` 等）以保持风格一致，而 **QMT**使用统一的 `passorder()` 函数，需要传入委托类型、交易标的、价格、数量等一系列参数<sup>8</sup>。上述差异意味着原策略的下单逻辑需重写：如将 `order_target(security, 0)` 卖出指令替换为 QMT 的 `passorder(24, 1101, account, security, 5, -1, quantity, C)` 参数组合形式<sup>9</sup> <sup>10</sup>。这部分改动较繁琐，需要查阅QMT官方文档确定每个参数的含义（例如23表示市价买，24表示市价卖等）。

• **账户和持仓：**聚宽通过 `context.portfolio` 获取持仓、市值和现金等信息，例如 `context.portfolio.positions` 是持仓字典，`context.portfolio.cash` 是可用资金。在QMT 中没有直接等价的 `context` 对象，需要改用其账户查询接口：如 `get_trade_detail_data(account, "stock", "position")` 返回持仓列表，`get_trade_detail_data(account, "stock", "account")` 返回账户资金信息<sup>11</sup> <sup>12</sup>。因此迁移时，凡是 `context.portfolio` 相关用法需重写为QMT的数据查询调用，并可能自行维护一个持仓字典以便逻辑判断<sup>12</sup>。PTrade则延续了聚宽的 `context` 概念，提供类似的持仓与现金获取方式，使这部分迁移较为直接。

· **行情数据获取**：聚宽提供便捷的行情函数如 `attribute_history / get_bars` (获取一段历史K线) 和 `get_current_data` (实时数据) 等。PTrade由于与聚宽同源，这些API大多直接可用或名字略有不同，如获取历史数据可能用 `history` 或 `get_price` 接口。QMT的数据接口则需要通过 `ContextInfo` 对象的方法，例如 `C.get_market_data_ex(fields, stock_list, period, end_time)` 来获取行情<sup>13</sup>。举例来说，聚宽中的 `history = attribute_history('000001.XSHE', 5, '1d', ['close'])` 相当于 QMT 中：先设定标的列表 `C.stock_list=["000001.SZ"]`，然后在 `handlebar` 中调用 `market_data = C.get_market_data_ex(["close"], C.stock_list, period="1d", end_time=bar_time)` 获得字典，再从中提取收盘价序列计算均线<sup>14</sup><sup>15</sup>。需要注意 QMT 获取数据前需确保本地已下载相应行情数据；而PTrade在云端则由系统实时提供行情（可直接调用API获取）。此外，一些聚宽特有的方便函数（如 `get_current_data()` 获取实时快照、`get_today_data()` 等）在QMT上没有直接对应，需要通过订阅行情或连续调用实现。**值得庆幸的是，PTrade上许多接口与聚宽一致或高度相似**<sup>1</sup>，这使得从聚宽移植到PTrade主要是小范围修改，而移植到QMT往往改动较大、测试调试工作量更多。

**策略逻辑与细节差异**：除了API本身，不同平台对于策略细节也有区别。例如：

- 聚宽允许使用全局变量 `g` 来保存跨周期的信息，PTrade同样支持全局变量机制，而QMT则通常通过 `C` 上下文对象属性或Python全局变量来保存状态。迁移时应确保这些变量初始化和调用方式符合目标平台的作用域规则。
- 时间函数和日期处理：聚宽策略常用 `context.current_dt` 获取当前回测时间，PTrade可能也提供类似属性；QMT则需要使用 `C.get_bar_timetag()` 转换为日期时间<sup>16</sup>。凡是策略中涉及当前日期时间的逻辑（如判断月初月末、节假日等）需调整为目标平台支持的方式。
- **调仓/盘前盘后逻辑**：聚宽有 `before_trading_start` (开盘前) 和 `after_trading_end` (收盘后) 钩子函数。PTrade同样提供了这些接口<sup>17</sup>；QMT由于策略在本地运行，一般通过在 `handlebar` 内自行判断时间或利用定时任务插件实现每日一次的盘前/盘后动作。如果策略依赖盘前选股、数据预处理，那么在QMT上需要额外编写启动时或每日首次执行时的逻辑，以模拟聚宽的行为。

综上，**代码迁移在技术上是可行的，但难点在于API替换和策略框架调整**。简单策略（例如仅用日线行情和基本下单）的迁移相对容易，复杂策略（使用了聚宽特有诸多函数、调度频繁、第三方库）则需要较多修改<sup>18</sup>。在极端情况下，如果策略对平台耦合严重，**重构可能比逐个替换更省力**<sup>18</sup>。开发者应充分阅读目标平台的API文档，对照聚宽接口逐一替代，并在仿真环境反复测试，确保迁移后的策略逻辑和原策略一致。

## 2. 可用的迁移工具与框架

手工逐行修改代码虽然可行，但工作量大且易出错。好在社区和业界已出现一些工具和开源项目，能够帮助自动转换或兼容聚宽的策略：

- **BulletTrade 开源框架**：一个专为A股量化交易开发的本地一体化框架<sup>19</sup>。它完全兼容聚宽API和策略范式，提供了完整的回测与实盘方案，使你的聚宽策略代码几乎无需修改即可本地跑回测和实盘<sup>20</sup>。使用方式是在本地Python环境中安装 `bullet-trade` 包，然后将聚宽策略代码导入运行；BulletTrade内部实现了聚宽常用函数（如 `order`, `attribute_history`, `get_fundamentals` 等）的等价功能，支持多种数据源以及实盘下单接口<sup>21</sup>。对于希望脱离聚宽云平台、在自己环境延续使用原有策略的开发者，这是一种选择。需要注意BulletTrade本质上是一个独立框架，并非直接将代码迁移到券商平台，而是通过兼容

层实现本地化。如果你的目标是对接券商实盘（如国金QMT、恒生PTrade等），BulletTrade也支持接入券商交易接口，实现下单。<sup>22</sup>

- **一键代码转换工具：** 目前已经有开发者利用大语言模型（LLM）构建了自动转换工具。例如 GitHub 上的 **Quant2PTrader-MCP** 项目提供了一个基于 Model Context Protocol (MCP) 的服务，用于将聚宽策略代码自动转换为 PTrade 格式代码<sup>23</sup>。其核心功能包括：输入完整聚宽代码可一键输出对应的PTrade代码，自动识别所有需要替换的API，并生成详细的转换报告（标注哪些部分已替换，哪些可能需手工调整）<sup>24</sup>。该工具还强调了高兼容性，尽量实现两平台API的无缝映射<sup>25</sup>。类似地，有社区网友搭建了**在线AI转换网站**，通过分析聚宽策略逻辑后按照PTrade接口规范重写代码，实现“一键转换”<sup>26</sup>。这些工具对于标准化的策略（不涉及过于复杂的自定义操作）效果较好，可以节省大量机械替换工作。不过自动转换的代码仍需人工审核和调试：转换工具会给出**风险提示**，提醒开发者哪些地方可能需要进一步修改<sup>25</sup>。
- **聚宽信号转发实盘方案：** 除了直接转换代码，还有一种思路是“**曲线救国**”：利用聚宽模拟盘产生交易信号，再通过工具将信号转给实盘系统执行。这类方案绕过了代码层面的迁移，典型实现是“**聚宽+QMT跟单系统**”<sup>27</sup>。其原理是：在聚宽策略中嵌入一段发送信号的代码（例如调用接口将订单指令发送到本地服务器或 Redis 消息队列），然后在本地运行一个QMT脚本监听该信号并调用 QMT 下单<sup>28 29</sup>。换言之，**QMT端只负责执行交易，不重新计算策略**。有开源项目提供了现成的聚宽跟单脚本（如将 TraderLink.py 上传至聚宽研究环境，并在策略开头导入）和 QMT端订阅程序，实现每隔几秒拉取聚宽信号，有交易指令时自动提交券商<sup>30 31</sup>。这种方式的优点是不需改动原策略逻辑，快速实现实盘；缺点是架构较复杂，对网络和同步要求高，信号延迟和丢失需要谨慎处理<sup>29</sup>。**QMT方案的核心思想就是绕开代码转换**，直接捕捉聚宽策略运行时生成的模拟交易信号，再转化为实盘委托<sup>32</sup>（而PTrade方案更倾向于“代码重构”直接在平台运行<sup>32</sup>）。如果你的策略在聚宽上已经稳定运行且不易改动，这是可以考虑的权宜之计。
- **Mini QMT 与 XTQuant：** 官方层面，迅投QMT提供了 **MiniQMT 模式**，允许部分券商用户使用原生 Python 环境连接QMT<sup>2</sup>。通过 XTQuant 接口，策略主要逻辑可以在独立的Python脚本中运行，仅将交易指令发送给QMT作为通道。开发者可以因此利用任何第三方数据源（如 Tushare 财经数据）和熟悉的开发框架，只把QMT当成**下单柜台**<sup>2</sup>。这种模式本质上也需要对接QMT的API，但**自由度更高**：你甚至可以把聚宽的API封装成自己的函数，在XTQuant调用券商交易，实现类似聚宽的体验。因此，MiniQMT提供了一种官方支持的“兼容层”思路——通过**API封装**将你的策略适配到QMT执行。在PTrade方面，由于是云端平台，不支持类似的本地运行模式，但PTrade可以通过REST API（若提供）与外部通信，实现信号对接等功能（需查阅各券商对PTrade的开放接口支持）。
- **其他开源项目：** 业内还有一些相关项目值得关注。例如 **SimTradeLab**<sup>33</sup> 是一个参考PTrade架构的开源回测框架，实现了与PTrade API语法100%兼容的本地环境，可用于在迁移前测试策略的行为与PTrade一致。再如很多量化社区分享的**实盘策略模板**，包含典型的聚宽转PTrade/QMT代码示例和常见坑点处理（如在知乎、掘金、CSDN等平台上一些教程帖子）。这些资源有助于开发者**对照修改**：通过模板学习如何调用券商数据、如何编写QMT的下单指令等，从而举一反三迁移自己的策略。

### 3. 自动转换工具链的开发思路

如果现有工具无法满足需求，开发者也可以考虑**自建一套策略迁移/兼容工具链**。以下是几种可行的思路：

- **策略DSL中间层：** 抽象出一套独立于平台的策略描述语言或配置（类似于领域专用语言）。开发者用该中间层编写策略逻辑，例如定义买卖信号、调仓频率、选股条件等。然后为每个目标平台开发对应的“**代码生成器**”，将中间层翻译成平台代码。例如定义 `BUY_SIGNAL = price > MA5`，生成聚宽代码时转成 `if`

`current_price > MA5: order(...)`，生成QMT代码则转成等价的 `passorder(...)` 调用。通过这种编译器式的设计，未来新增平台只需扩展一个代码后端。难点在于准确覆盖所有策略逻辑——DSL需要足够灵活表达各种量化策略，否则复杂策略仍难以用简单描述表示。此外，维护DSL与各平台API的同步也是一项持续工作。

- 兼容API封装层： 和BulletTrade类似，可以编写一个在目标平台上运行的适配库，让它提供聚宽风格的API，再在策略代码前引入该库以支撑原有调用。举例来说，在QMT本地策略代码里先`import JQCompat`，其中定义了 `order(security, amt)` 函数内部调用QMT的 `passorder` 实现下单，定义 `attribute_history` 函数内部调用 `C.get_market_data_ex` 获取历史数据并返回类似 DataFrame结果。这样，理论上无需修改策略主体，只要保证所有聚宽API都有对应实现即可。在QMT上，由于可以使用任意Python库，此路可行；在PTrade云端则可能受限（未必允许用户加载自定义模块）。即便PTrade限制加载第三方库，也可以尝试将兼容代码直接插入策略文件开头。当封装充分完善时，聚宽的大部分代码都能直接运行。不过要留意性能和限制：比如QMT获取数据需要预先下载，封装层应处理好异常；再如聚宽的某些接口（如可视化 `record`）在实盘无意义，可选择性跳过或模拟。
- 模块化分离策略逻辑： 在迁移过程中，建议将策略划分为平台无关的核心逻辑和平台相关的接口层。核心逻辑包括选股算法、信号判断、仓位计算等，这部分代码尽量避免直接调用平台API，可以用纯Python实现并通过参数传递数据。接口层则负责数据提取（如读取行情、财务数据）和交易执行（如提交订单）。当初始策略 tightly 耦合平台时，需要重构出这样的分层结构。完成后，针对不同平台编写不同的接口模块，但重用相同的核心逻辑模块。例如实现一个 `DataProvider` 类，聚宽版调用聚宽API，QMT版调用QMT API；再实现一个 `BrokerExecutor` 类，封装下单操作。同一套逻辑通过不同实现的接口类注入，即可在多平台运行。这种策略模式的代码组织，有助于日后同时维护多份策略版本：当核心逻辑需要调整时，各平台表现保持一致，而平台API的差异由各自接口模块处理。
- 多平台版本管理： 如果需要长期维护策略在多个平台上的实现，必须有良好的版本管理策略。一种做法是采用单一代码仓库，多配置构建：使用预处理或模板技术，将公共部分和差异部分统一管理。例如用宏或标志将不同平台代码片段隔离，同一源码通过配置生成适配聚宽或QMT的版本。这有点类似于跨平台软件开发中的多目标构建。另一种简单方式是Git分支管理：以聚宽版本为基准，新建分支修改为QMT版、PTrade版，之后尽量在公共逻辑上保持同步，每当策略逻辑更新时手动合并到各分支。这要求开发者严格记录每次修改，并仔细测试各版本结果一致性。借助自动化测试也很重要：可以针对历史行情和信号，编写测试用例分别跑在聚宽、PTrade、QMT版本上，检验输出是否一致，从而及时发现迁移问题并修正。

总的来说，自动转换或多版本协同的关键在于抽象。要么向上抽象出统一的策略描述，让机器去适配不同平台；要么向下抽象出统一的接口层，在策略和平台间搭建桥梁。实际应用中，两者可以结合：既利用工具自动初步转换，再由人工调整封装层，最终形成可持续维护的多套策略代码。

## 4. 数据调用与选股逻辑的迁移适配

财务数据接口转换：很多量化策略涉及基本面选股，例如用聚宽的 `get_fundamentals()` 查询财务指标、市值等筛选股票。迁移时需要将这部分改用目标平台的数据源：

- 在PTrade上：好消息是PTrade直接提供了 `get_fundamentals` 接口，功能与聚宽类似<sup>34</sup>。它支持查询上市公司三大财务报表、日频估值数据和各种财务指标，数据源为恒生聚源等专业数据<sup>34</sup>。调用方式也与聚宽接近，例如：`get_fundamentals(security_list, 'valuation', fields=['pe_ratio', 'pb_ratio'], date='20231231')` 可获取多只股票在指定日期的PE、PB。

需要注意流量控制（每秒调用次数有限）<sup>35</sup> 以及季度数据的特殊性（只能查询已发布财报的季度）<sup>36</sup>。因此，将聚宽的财务数据查询迁移到PTrade基本是直接替换函数名，稍作参数调整即可完成。

· 在QMT上：

QMT也提供了财务数据查询功能，但方式有所不同。QMT中的 `ContextInfo.get_financial_data(field_list, stock_list, ...)` 可获取财务字段数据<sup>37</sup>。不过使用前必须先下载离线财务数据库（通过QMT客户端的数据下载功能），否则调用将返回空<sup>37</sup>。一旦数据下载完毕，你可以在策略中使用该接口按需提取财务指标。举例：在init阶段调用 `C.get_financial_data(['net_profit', 'total_assets'], ['000001.SZ'], report_date='2023Q3')` 来获取指定股票特定季度的净利润和总资产。QMT对财务数据还提供了一系列特色函数<sup>38</sup>，比如获取十大股东、行业分类等。如果聚宽策略用到了类似 `get_industry()` 等函数，QMT可以通过其**行业概念数据**接口找到对应办法。总体而言，在QMT实现基本面选股需要多一步“**准备数据**”的动作，并习惯其基于本地数据库的查询方式。

· 第三方数据源替代：有时候，目标平台自带的数据可能不全或不便使用。例如一些聚宽策略用到了特殊的因子或数据（如聚宽的 alpha191 因子库、新闻情绪数据等），PTrade/QMT未必提供。这种情况下，可以考虑使用**开源数据接口库**（如 Tushare、AkShare、Yahoo财经API 等）来替代聚宽的数据来源。尤其在QMT中，由于运行在原生Python环境，调用第三方数据非常灵活。在MiniQMT模式下，不仅可以获取外部数据，还能结合pandas等库进行复杂选股计算，然后通过QMT下单。采用外部数据时要注意**数据一致性**：尽量确保和聚宽使用的数据口径一致，以免策略逻辑因数据差异出现偏差。

**选股模块迁移**：聚宽提供了一些高级API用于选股和构建股票池，例如 `get_index_stocks` 获取指数成分股、`filter_query` 配合财务指标筛选股票等。PTrade大多提供了对应功能，如获取指数成分、行业分类的函数，以及可结合 `get_fundamentals` 手动筛选。QMT则需要开发者**自行实现选股逻辑**：例如通过获取某一板块列表，再逐只调用 `get_financial_data` 或行情数据筛选。**业界有一些成熟方案**可以借鉴：比如利用QMT的因子平台或社区分享的选股脚本，将聚宽复杂的选股条件拆解成多个步骤执行。在迅投QMT知识库和社区论坛上，常见范例包括**多因子选股策略**实现教程<sup>39</sup>，演示了如何用QMT提取多只股票的财务数据并排序选出投资组合。另外，如果聚宽策略依赖定期调仓（如每月末选一次股），在QMT中可以通过在 `handlebar` 判断日期来触发选股函数，每到调仓日更新股票池列表。

**案例与方案**：假如原策略使用 `get_fundamentals` 获取低市盈率且盈利增长为正的股票列表，然后每月买入前N只股票。那么在PTrade上，可以几乎照搬这段代码（改成PTrade接口调用）来得到股票列表<sup>34</sup>；在QMT上，建议在策略初始化时或定时点，调用 `C.get_financial_data` 分别获取全部候选股票的PE和净利润增长率数据，然后用pandas筛选排序得到目标股票池，最后对当前持仓调整实现调仓。虽然步骤多一点，但结果应和聚宽一致。值得关注的是，QMT由于本地运行，会受限于本地硬件性能和数据规模，如果一次查询上千只股票所有财务数据，可能比较慢或占用内存，需合理安排（如分批获取或提前缓存数据）。另外，在**实盘中基本面数据更新频率较低**（季度报），因此可以考虑将财务选股过程放在盘后或特定日期运行，而不必每个交易日都执行，提高效率<sup>40</sup>。

总之，**选股和数据处理部分的迁移重点在于找到对应的数据接口**。PTrade与聚宽差异较小，一般是API名称和参数的对照；QMT则需要熟悉其本地数据查询方式，必要时结合外部数据源。开发者可以查阅目标平台的API文档中“**数据**”部分：例如迅投知识库的“股票数据”“财务数据接口”章节<sup>41</sup>、PTrade的API手册<sup>42</sup>等，了解各种数据获取函数的使用方法。在迁移过程中，多比较几只股票的小样本数据确保新平台取到的数据正确无误，再逐步还原整体选股逻辑。

## 5. 参考资料与迁移资源

为了帮助开发者更快上手，这里汇总了一些教程、工具和文档资源，可作为聚宽迁移到PTrade/QMT的参考：

- **PTrade 官方API文档**：<sup>43</sup> PTrade提供详细的量化交易接口文档，包括策略框架、交易函数和数据函数说明。如 [ptradeapi.com](http://ptradeapi.com) 网站列出了各API用法（初始化、handle\_data、get\_fundamentals等）。通过该文档可以逐项对比聚宽和PTrade的API差异，指导代码修改。
- **QMT迅投知识库**：官方的[迅投QMT知识库](#)包含“聚宽策略迁移至QMT”的专门指南<sup>44</sup>。其中对比了聚宽与QMT的代码结构，并给出一个5日均线策略从聚宽版本到QMT版本的完整转换示例<sup>45</sup><sup>9</sup>。知识库还详细列出了QMT特有功能（如订阅实时行情、获取期权列表等<sup>38</sup>），是学习QMT编程和迁移策略的必读资料。
- **BulletTrade 项目文档和代码仓库**：<sup>22</sup> BulletTrade的官方站点（[bullettrade.cn](http://bullettrade.cn)）提供了教程，包括“三步跑通回测/实盘，聚宽策略无需改直接复用”指南<sup>46</sup>。其GitHub仓库<sup>47</sup>公开了实现细节。如果考虑使用BulletTrade实现本地实盘，可以参考其文档了解如何配置数据源、对接券商交易，以及查看示例策略代码在BulletTrade上的写法。
- **聚宽转PTrade 自动转换工具**：GitHub上的[Quant2Ptrader-MCP 项目](#)已开源，可用于一键转换聚宽代码到PTrade格式<sup>23</sup>。项目README介绍了使用方法，包括通过Claude等AI助手调用MCP服务完成代码转换的步骤<sup>48</sup><sup>49</sup>。这对于有一定编程基础并愿意尝试AI辅助开发的用户是一个有价值的工具。
- **实盘跟单方案教程**：知乎专栏文章「两行代码解决聚宽不能实盘的问题」详解了如何利用QMT捕捉聚宽交易信号实现实盘跟单<sup>32</sup>；CSDN博文《记录一次聚宽打通QMT的量化实践》提供了示例代码，演示如何用Redis作为介质传递信号<sup>28</sup>。另外，迅投社区论坛也有帖子分享了“聚宽跟单QMT脚本”的使用方法<sup>50</sup>。这些资料适合对信号转发方式感兴趣的开发者参考实践。
- **迁移经验分享文章**：一些博主和量化从业者分享了聚宽迁移的心路。例如：「聚宽停止服务，如何无痛转到PTrade平台？」<sup>51</sup>一文讨论了聚宽和PTrade在界面和API上的高度相似，以及切换到PTrade所需的少量代码改动。<sup>52</sup>而知乎问答「聚宽的策略怎么移植到QMT？」下有答主给出了分步骤指导，从拆分策略逻辑到替换API，再到调试运行的全过程。如果在迁移中遇到具体问题，可以搜索这些关键词，很可能已有人遇到并给出解决方案。
- **视频课程与演示**：B站上有多个相关视频，例如《两行代码迁移聚宽策略到PTrade》演示了通过简单桥接代码将聚宽策略接入PTrade实盘的过程（包括行情同步和下单设置）；迅投官方推出的[QMT教程视频](#)（在其知识库视频教程板块）则系统讲解了QMT策略开发，从入门到高级功能。通过观看视频，开发者可以更加直观地理解平台差异和操作步骤。
- **实盘策略模板代码**：在GitHub和量化社区中，也能找到一些PTrade或QMT实盘策略的示例仓库。例如有人将经典的均线策略、海龟策略等实现于QMT/PTrade，并开源了代码。这些模板代码展示了目标平台上完整的策略框架、配置和实现细节，对于学习如何组织迁移后的代码很有帮助。推荐检索关键词如“Ptrade 策略示例 GitHub”或进入迅投社区的“策略模板”板块获取资源。

综上，充分利用上述资源，结合本文提供的思路，开发者可以有条不紊地将策略从韬睿量化/聚宽迁移到PTrade与QMT实盘环境。先从接口差异入手逐一替换，借助工具减少机械工作，再通过测试验证策略一致性，最后参考他人经验优化完善。希望这些指导能帮助您快速跨越平台差异，实现策略在实盘中的稳定运行！<sup>1</sup><sup>6</sup>

- 
- 1 2 第一创业聚宽量化交易即将停止运营，更好的方案来了：来看看PTrade QMT！ - 吾要开户  
<https://kaihu51.com/quant/705>
- 3 5 8 9 10 11 12 13 14 15 16 38 44 45 聚宽策略迁移至QMT | 迅投知识库  
<https://dict.thinktrader.net/strategy/JoinQuant2QMT.html>
- 4 量化交易API 接口开通-ptrade 量化交易代码编写教程  
<https://kaihu51.com/quant/1426>
- 6 7 18 52 聚宽JoinQuant 量化交易平台写的小市值策略如何转移到恒生PTrade 或迅投QMT - 吾要开户  
<https://kaihu51.com/quant/703>
- 17 PTrade操作流程：运行框架、新建策略、回测实盘（附开通方式）  
<https://m.douban.com/note/876476430/>
- 19 BulletTrade · 聚宽兼容量化框架  
<https://bullettrade.cn/>
- 20 22 46 47 GitHub - BulletTrade/bullet-trade: 全流程交易系统，兼容聚宽代码。  
<https://github.com/BulletTrade/bullet-trade>
- 21 bullettrade - bigbencat - 博客园  
<https://www.cnblogs.com/bullettrade>
- 23 24 25 48 49 GitHub - guangxiangdebizi/Quant2PTrader-MCP: 一键转换聚宽策略到Ptrade平台的MCP服务  
器  
<https://github.com/guangxiangdebizi/Quant2PTrader-MCP>
- 26 【聚宽策略迁移PTrade】聚宽代码AI 一键转换 - 百果量化交流平台  
<http://82.156.218.246/article/detail.html?id=768&sort=new>
- 27 聚宽的策略怎么移植到QMT? - 知乎  
<https://www.zhihu.com/question/554809363>
- 28 本地QMT跟单聚宽策略代码剖析讲解 - CSDN博客  
[https://blog.csdn.net/qq\\_26742269/article/details/147117380](https://blog.csdn.net/qq_26742269/article/details/147117380)
- 29 聚宽策略如何高效跟单QMT实盘？ - ZOL问答  
<https://ask.zol.com.cn/x/31151636.html>
- 30 31 聚宽实盘跟单攻略（量化交易必看！） - 知乎专栏  
<https://zhuanlan.zhihu.com/p/1918682286305550906>
- 32 两行代码解决聚宽不能实盘的问题（附行情获取函数）  
<https://zhuanlan.zhihu.com/p/1961089331662942260>
- 33 kay-ou/SimTradeLab - GitHub  
<https://github.com/kay-ou/SimTradeLab>
- 34 PTrade获取财务数据原创 - CSDN博客  
[https://blog.csdn.net/m0\\_46603114/article/details/133868889](https://blog.csdn.net/m0_46603114/article/details/133868889)
- 35 ptrade从零开始学习量化交易第48期【ptrade获取股票信息之 ...  
[https://blog.csdn.net/GJ57017\\_wx/article/details/139652568](https://blog.csdn.net/GJ57017_wx/article/details/139652568)

36 ptrade内部问题收集及解决方法（四）持续更新

<https://zhuanlan.zhihu.com/p/714827451>

37 41 获取数据 - 迅投知识库

<http://docs.thinktrader.net/pages/fd9cbd/>

39 qmt量化交易策略小白学习笔记第23期【qmt编程之获取财务数据 ...

<https://blog.csdn.net/fanglue3705/article/details/139624077>

40 量化交易Ptrade软件使用过程中常见问题总结

<https://zhuanlan.zhihu.com/p/1971266176685441899>

42 财务数据的API接口说明

<https://ptradeapi.com/hub/data/finance.html>

43 Ptrade 量化交易API接口文档

<https://ptradeapi.com/>

50 聚宽跟单&QMT实盘\_python\_草根创作者 - 量化交易与投资社区

<https://quant.csdn.net/6874aa5bbb9d8e0ecec22b9d.html>

51 聚宽停止服务，如何无痛转到PTrade平台？ - 雪球

<https://xueqiu.com/5957717280/296984929>