项目 1：综合多种来源的数据

导言

金融研究通常需要从不同来源收集数据集。在许多情况下，源数据的格式不便于导入 Python 进行分析。本评估将帮助您掌握数据获取、清理和合并的基本技能。您将合并分布在多个文件中的股票价格信息，并以 JSON 文件的形式输出。我们要求您编写通用代码，以便轻松适应不同的文件格式。

一开始，编写通用代码可能是一项艰巨的工作。不过，练习编写能适应不同配置的函数对您最有利。这样做会使你的代码更健壮、更易于维护和升级。

为了帮助您，我们提供了一个 Python *脚手架*。这个名为 zid_project1.py 的文件包含了本项目所需的所有函数。每个函数都有详细的*文档*说明，描述了函数的作用、输入参数和返回对象。

本文件的其余部分将提供以下信息：

- 您将收到的数据文件。
- 在 PyCharm 中设置开发环境的说明。
- 完成评估所需的详细步骤说明。请严格按照这些说明进行操作。我们评估您作品的能力要求您这样做。

您应该在 PyCharm 中开发您的代码。不过，提交将通过 *Ed 进行*。您只需将 zid_project1.py 文件复制到 *Ed 中即可*。与您迄今为止完成的代码挑战不同，*Ed* **不会**对您的代码提供任何反馈。您仍然可以在截止日期前多次提交 - 只有您在 *Ed* 上的最后一次提交才会被标记。

源文件

所有所需文件都包含在一个 zip 文件夹中，其结构如下：

```
项目1
|    project_desc.pdf
|    README.txt
|    TICEKRS.txt
|    zid_project1.py          <-- 这是唯一需要提交的文件
|___data/
|    |    <tic>_prc.dat       <-- 此类型的多个文件
```

- zid_project1.py 包含您将在本评估中使用的函数。其中一些函数已经编写完成，另一些则需要您自己编写。有关详细信息，请参阅下面的说明。
- README.txt 文件包含股票数据如何存储在 .dat 文件中的信息。请使用您收到的 README.txt 文件中包含的信息（不同的学生有不同的版本）。
- TICKERS.txt 包含股票代码及其相应交易所的列表，每行一个。这些代号和交易所的大小写

均可。

- project_desc.pdf 是本文档的 PDF 版本。
- data 是一个子文件夹，其中包含完成评估所需的所有数据。在该文件夹中，您将找到许多文件。每个 <tic>_prc.dat 文件都包含股票代码 <tic> 的股价数据。TICKERS.txt 中的每个股票代码都有一个相应的".dat "文件。不过，TICKERS.txt 文件中的".dat "文件可能多于股票代码。换句话说，".dat "文件的数量可能多于您的需要（但也不会少于您的需要）。

说明

要为该评估设置 PyCharm，请按照以下步骤操作：

1. 将压缩文件的内容解压到电脑上。

2. 将整个 project1 文件夹复制/移动到 PyCharm 工具包项目文件夹中。之后，您的工具包文

```
工具包/                          <-- 项目文件夹
|    toolkit_config.py           <-- 已创建
|    ...
|___项目1/
|    |    project_desc.pdf       <-- 压缩文件夹的内容
|    |    README.txt
|    |    TICEKRS.txt
|    |    zid_project1.py
|    |
|    |___data/
|    |    |    <tic>_prc.dat
```

   件夹将看起来像这样：

3. 完成 zid_project1.py 中用户编写的函数。请参阅
   *完成*下面的*代码脚手架。*

4. 在 PyCharm 中完成 zid_project1.py 模块后，将该模块的全部内容复制并粘贴到 *Ed 中*。这是您完成评估所需提交的唯一文件。

5. 按 "提交 "键提交项目。在提交之前，您的项目不会被提交。

完成代码脚手架

用项目文件设置好 PyCharm 开发环境后（见上文说明），按以下步骤依次修改 zid_project1.py 模块。完成的代码将生成一个 JSON，其中包含多个文件的组合内容。您**无需**提交此 JSON 文件。

步骤 1：设置文件和文件夹的位置（5 分）

在 PyCharm 中打开 zid_project1.py 模块。

如下所述，为常量 ROOTDIR、DATDIR 和 TICPATH 设置正确的表达式。重要的是，在定义这些变量时，不能包含正斜杠或反斜杠（因此不能使用 "C:\User. . "等）。相反，你应该使用 os 模块中的相应方法。

- ROOTDIR 变量结合了工具包项目文件夹（已在 toolkit_config.py 模块中指定）和 project1 软件包的基本位置。
- DATDIR 变量结合了上述 ROOTDIR 中的位置和数据子文件夹。请注意，这与 toolkit_config.py 模块中的 DATADIR 变量不同。DATDIR 变量的名称是 "DAT "和 "DIR "的组合（不是 "DATA "+"DIR"），它指向计算机中的不同位置。

- TICPATH 变量结合了 ROOTDIR 中的位置和包含 tickers 的文件名 (TICKERS.txt)。

同样，所有这些路径都应使用 os 模块中的相应方法创建。如果在这些变量的定义中包含任何正斜杠或反斜杠，你的代码将只能在你的计算机上运行。本次评估的一个重要部分就是确保代码的可移植性。

下图显示了这些变量及其位置之间的关系。再次强调，不要使用完整路径（如 "C:\Users. . ."）

```
工具包/
|    toolkit_config.py          <-- 您已创建此模块
|    ...
|___项目1/                      <--`ROOTDIR` 变量指向该文件夹
|    |    ...
|    |    TICEKRS.txt           <--`TICPATH` 变量 | 指向此文件
|    |    ...
|    |    zid_project1.py       <--`DATDIR` 变量 | 指向此文件夹
|    |___data/
|    |    |    ...
```

来创建这些变量。

注意：这部分只需用适当的表达式替换"<完成本部分>"字符串即可。

步骤 2：设置描述源数据格式的变量（5 分）

这部分内容非常重要！请务必严格遵守这些说明。

开始之前，请在 PyCharm 中打开一个 ".dat " 文件。为此，只需导航到 toolkit/project1/data/ 文件夹（在 PyCharm 中），然后双击其中一个文件。您会发现如下内容：

1. 没有列头。文件中的每一行（包括第一行）都包含数据。
2. 各栏之间没有 "分隔符"（例如，各栏之间没有逗号、制表符等分隔符）。
).例如，前 10 个字符属于第 1 列，后 8 个字符属于第 2 列，等等。

这意味着我们必须创建一个函数，将行分割成列，以便将每个 "值 "分配给正确的 "数据字段"。第一步是为变量 COLUMNS 和 COLWIDTHS 设置正确的表达式。

- COLUMNS 变量必须是一个列表，其中每个元素代表 README.txt 文件中的一个源列名。列表中元素的顺序必须与 README.txt 文件中列的顺序一致。例如，假设 README.txt 文件中有以下信息：

  ```
  # ----------------------------------------------------------------------------
  #   列信息 #
  # ----------------------------------------------------------------------------
  关闭：
    立柱位置：1
    宽度： 14 日
  期
    栏位：2
    宽度：11
  ```

  在这种情况下，必须设置 COLUMNS = ['Close', 'Date']。

- COLWIDTHS 变量必须是一个字典。每个键都是 COLUMNS 中的列名。每个值是 README.txt 文

件中该列的宽度。在上面的示例中，你将设置 COLSWIDTH
= {'收盘'： 14，'日期'： 11}。

步骤 3：完成 get_tics 函数（15 分）

完成函数 get_tics 的指定部分。该函数读取一个包含股票代码的文件，并返回一个包含格式化股票代码的列表。请确保该函数使用给定的 pth 变量而**非** TICPATH 常量（即该函数中不应有对 TICPATH 常量的引用）。我们将使用不同的文件来测试你的代码。在 get_tics 中使用 TICPATH 而不是 pth 意味着您的函数将始终返回相同的股票，而不是适应不同的股票列表。

您的函数必须与所提供的*文档字符串*一致。特别是，请确保函数正文与文档中的 "参数 "和 "返回值 "部分保持一致。唯一的例外是 "提示 "部分提供的可选建议，您无需遵循。

模块中还有一个名为 _test_get_tics 的测试函数。创建完 get_tics 函数后，最好运行 _test_get_tics 函数并查看输出结果。这应该能很好地说明函数是否按预期运行。您可以取消 if name.....代码块的相关部分，以运行该测试函数。与提供的所有其他测试函数一样，您可以
可以修改或删除这些功能 - 它们不会被标记。

步骤 4：完成 read_dat 函数（15 分）

完成函数 read_dat 的指定部分。该函数读取给定股票代码的股价数据文件，并以行列表的形式返回其内容。请确保函数正文与 docstring 的 "参数 "和 "返回值 "部分一致。您可以选择遵循 "提示 "部分的建议步骤，但这是可选的。

切记不要在函数正文（或模块中的任何地方）中使用带有完整路径的字面形式，如 "C:\Users. ... " 等字面意义的完整路径。你可以使用上文第 1 步创建的常量（如 DATDIR）和 os 模块中的方法来创建路径。 您可以使用相应的 "test "函数
_test_read_dat，以便在函数完成后对其进行测试。

步骤 5：完成 line_to_dict 函数（15 分）

完成函数 line_to_dict 的指定部分。与
上述 get_tics 和 read_dat 函数也适用于该函数。

步骤 6：完成 verify_tickers 函数（10 分）

完成函数 verify_tickers 的指定部分。该函数接收待验证的股票代码列表，如果提供的任何股票代码不是 get_tics 函数返回的字典中的键，则引发异常。关于何时引发异常的更多详情，请参阅 *docstring* 的 "注意事项 "部分。

异常是一种扰乱程序正常运行的行为。这种操作通常代表了错误的抛出。异常是我们优雅地从错误中恢复的方法。

要了解有关提出例外情况的更多信息，您可以参考以下资源：
https://www.w3schools.com/python/gloss_python_raise.asp

步骤 7：完成 verify_cols 函数（10 分）

完成函数 verify_cols 的指定部分。该函数接收要验证的列名列表，如果提供的列名在 COLUMNS 中找不到，则引发异常。关于何时引发异常的更多详情，请参阅 *docstring* 的 "注意事项 "部分。

步骤 8：完成 create_data_dict 函数（20 分）

完成函数 create_data_dict 的指定部分。该函数用于将".dat "文件中的数据转换为单个字典。

该函数接收 3 个参数：

1. tic_exchange_dic
   - 函数 get_tics 返回的字典。
2. 代码_lst
   - 一个列表，其中包含我们要将其数据保存到由
     创建数据文件。
3. col_lst
   - 我们希望保存在由
     创建数据文件。

下面举例说明当我们调用
下面提供 create_data_dict(tic_exchange_dic,['aapl', 'baba'], ['Date', 'Close']):

```
{
    aapl': {
        交易所": "纳斯达克
        ", "数据": [
            {
                日期": "2020-01-01"、
                关闭": "8.0927"、
            },
            {
                日期": "2020-01-01"、
                关闭": "8.2784"、
            },
            ---
        ]
    },
    巴巴": {
        交易所": "纽约证券
        交易所", "数据": [
            {
                日期": "2017-05-13"、
                关闭": "3.4939"、
            },
            {
                日期": "2017-05-14"、
                关闭": "3.5689"、
            },
            ---
        ]
    }
}
```

- 行情数据列表中的每个字典应只包含 col_list 指定的列。
- 每个代号的数据列表应包含该代号".dat "文件每一行的字典。

注：上述示例中使用的数字完全是任意的，仅供参考。

步骤 9：完成 create_json 函数（5 分）

完成函数 create_json 的指定部分。该函数将给定的字典保存到 JSON 文件中。要进一步了解如何将数据写入 JSON 文件，请查看内置 Python 包 json 中下列方法的文档：

- json.dump
- json.dumps

提交您的模块

将 zid_project1.py 的全部内容复制并粘贴到 *Ed*，然后按 "提交"。*Ed* 不会自动标记您的项目，也不会给您任何反馈。提交后，我们就可以标记您的项目了。

管理指南、补充提示、标记

行政指南

我们将执行以下规定：

1. 本评估必须单独完成。如果不能独立完成作业，可能会被扣满分。
2. 允许逾期提交，但将根据课程大纲中所述的规则予以处罚。

提示

您的代码应该是可移植的，能在各种环境下运行。将您的代码从 PyCharm 复制到 *Ed* 提交即可。但是，作为评估的一部分，您需要确保您的代码在我们的电脑和您的电脑上都能运行。

以下提示可帮助您纠正任何可移植性错误：

1. zid_project1.py 模块的内容**不得包含对计算机中文件夹的任何直接引用**。定义路径的变量**不应**包含任何正斜杠或反斜杠。当然，您在 toolkit_config.py 模块（无需提交）中定义的变量会包含正斜杠或反斜杠（取决于您的操作系统）。这也是我们创建该文件的原因之一。
2. 同样，您也**不应在** zid_project1.py 模块中包含包含特定代号的字符串（如 "TSLA"、"AAPL"）。例如，您不应创建名为 tickers 的变量，然后将收到的特定代号复制到 TICKERS.txt 文件中。相反，您的代码应该读取 TICKERS.txt 文件，生成一个股票代码列表，并将其存储在一个变量中。
3. 在文件 zid_project1.py 中编写函数时：

- 请勿修改函数名称或参数。
- 只修改"<COMPLETE THIS PART>（<完成此部分>）"标记所指示的部分。
- 您无需导入任何其他模块。请不要修改导入语句。
- 您不应创建任何其他常量。文件（ROOTDIR、DATDIR、TICPATH、COLUMNS 和 COLWIDTHS）应按说明进行编辑。
- 包含的 "测试 "函数可帮助您在整个项目中测试代码。这些函数不会被标记，您可以根据需要对其进行修改。文件中以 _test 开头的名称清楚地标识了测试函数。

4. 在函数声明中使用所有参数。例如，函数 get_tics(pth) 只有一个参数 pth。请确保您的函数使用的是该参数，而不是全局变量。
5. **只**提交 zid_project1.py 模块。请确保您的代码仅适用于该模块。不能提交其他模块。

我们将如何为您的评估打分

本次评估将对以下部分进行评分：

1. 文件和文件夹的位置（5 分）
2. 设置描述源数据格式的变量（5 分）
3. 完成 get_tics 函数（15 分）
4. 完成 read_dat 函数（15 分）
5. 完成 line_too_dict 函数（15 分）
6. 完成 verify_tickers 函数（10 分）
7. 完成 verify_cols 函数（10 分）
8. 完成 create_data_dict 函数（20 分）
9. 完成 create_json 函数（5 分）

要获得第 1 和第 2 部分的全部学分，您的变量必须

1. 具有正确的类型（例如，COLWIDTHS 必须是一个字典）

2. 包含正确的值（例如，COLUMNS 中列的顺序必须与 README.TXT 文件中指定的顺序一致）、
3. 请遵守本文件中的所有说明（例如，在 TICPATH 中不得使用正斜线或反斜线

）。要获得第 3 - 9 部分的满分，您的函数必须

1. 返回正确的对象类型（详见文档说明）
2. 从收到的 TICKERS.txt、README.txt 和".dat "文件中返回正确信息。
3. 不违反我们在本文档或文档字符串中规定的任何规则
4. 如果函数打开文件，则**必须使用上下文管理器**。