Testing Momentum Strategies using Python

Bachelor's Thesis

Submitted in partial fulfillment of the requirements for the degree of
Bachelor of Arts in Banking and Finance

Author
Saskia Senn
Rainackerstrasse 6, 8953 Dietikon
19-734-664
saskia.senn2@uzh.ch

Chair of
Prof. Dr. Markus Leippold
Professor of Financial Engineering
Department of Banking and Finance
University of Zurich

Supervisor
Dr. Benjamin Wilding

Date of Submission: March 1, 2023

Abstract

The aim of this thesis is to develop an automated correction tool using the Python programming language to efficiently correct the *Involving Activity 3* in the course *Asset Management: Investments*. The exercise requires students to create two momentum strategies, a long-only and a long-short, based on historical stock prices of 18 stocks using varying look-back and holding periods. The tool is designed to be highly flexible in terms of input data, lock-back, and holding periods, enabling the momentum strategies to be effectively tested and compared to a buy-and-hold strategy. The tool offers a powerful approach for correcting the *Involving Activity 3* leading to faster processing times and minimized errors compared to manual correction methods.

Institut für Banking und Finance

Universität Zürich
Institut für Banking und Finance
Plattenstrasse 14
CH-8032 Zürich
Telefon +41 44 634 29 51
www.bf.uzh.ch

Bachelorarbeitssauftrag für:
Saskia Senn
Oberwiesenweg 26
5436 Würenlos

Prof. Dr. Markus Leippold

Prof. Dr. Markus Leippold
Professor of Financial Engineering

Zürich, 21. Juli 2022

Auftragserteilung für die Bachelorarbeit

Sehr geehrte Frau Senn

Mit Freude habe ich erfahren, dass Sie bei mir die Bachelorarbeit schreiben möchten. Hiermit erteile ich Ihnen dafür den folgenden Auftrag:

Titel: Testing Momentum Strategies using Python

Zielsetzung:
In the first, short part of your thesis, develop the theoretical foundations as well as an overview of the existing scientific literature - especially empirical studies - on momentum strategies.

In the second part, develop a tool in Python that can be used to test momentum strategies. Make sure that the tool is designed to be flexible with respect to input data and lockback and holding periods. Comment your code in detail.

Finish your work with a conclusion.

Ablauf Ihrer Arbeit

- Sie erhalten die Möglichkeit, eine allfällig erstellte Disposition mit Ihrer Betreuungsperson am Institut zu besprechen. Machen Sie dazu mit Ihrer Betreuungsperson einen Termin aus und senden Sie dazu Ihre Disposition.

Bitte beachten Sie folgende formale Kriterien:

- Sie verfassen die Arbeit in deutscher Sprache.
- Qualität geht vor Quantität. Achten Sie auf kompakte Schreibweise; verzichten Sie auf lange und unnötige Ausführungen und kommen Sie rasch zum Wesentlichen. Ihre Arbeit (exkl. Verzeichnissen und Anhang) sollte maximal 40 Seiten umfassen.
- Achten Sie bitte auf korrekte, fehlerfreie Sprache und einen wissenschaftlichen, knappen, aber flüssigen Schreibstil. Achten Sie auch auf korrekte Zitierweise.
- Verwenden Sie ausreichend beschriebene Graphiken. Graphiken und Tabellen sollten selbsterklärend und unabhängig von der Arbeit verständlich sein.
- Bauen Sie Ihre Arbeit wie folgt auf:
    - Titelblatt
    - Abstract nach dem Vorbild wissenschaftlicher Zeitschriften (max. 100 Worte)
    - Vorliegende Auftragserteilung
    - Inhaltsverzeichnis
    - Hauptteil
- Lade auf OLAT ein ZIP-File mit folgenden Dateien:
    - Titelblatt der Arbeit (PDF-Dokument)
    - Executive Summary (PDF-Dokument): Zusammenfassung auf max. 3 Seiten
    - Gesamte Arbeit (LaTex- oder Word-File sowie PDF-Dokument)
    - Gesamtes verwendetes Datenmaterial
    - Computer-Codes zur Replikation Ihrer Ergebnisse
    - Elektronisch gespeicherte Referenzen (Papers in PDF-Format).

Bei allfälligen Fragen wenden Sie sich an Benjamin Wilding, Institut für Banking und Finance,
Email: benjamin.wilding@bf.uzh.ch

Gute Arbeit und viel Erfolg!

Freundliche Grüsse

Markus Leippold
Professor of Financial Engineering
Universität Zürich
Institut für Banking und Finance

Contents

List of Tables

List of Figures

1 Introduction

"The trend is your friend", is a famous quote when it comes to momentum investing. Momentum investing is a strategy that involves buying past winners and selling past losers (Jegadeesh and Titman (1993)). The idea of momentum investing dates back to Levy (1967) who initially considered stocks that historically(called look-back period) have been relatively strong tend to remain relatively strong for a significant period of time (called holding period). Although, this idea contradicts one of the most important approaches in modern finance, the Efficient Market Hypothesis. Whereas, this hypothesis claims that capital markets are efficient and all information is included in the price (Fama (1970)). Thus, implying that it shall not be possible to gain superior returns over the market.

According to Fama (1970), a market in which prices always fully reflect all available information is called efficient. There are three stages of an efficient market. Firstly, the weak form focuses on a limited subset of information, namely, past price or return histories. Secondly, the semi strong form in which prices seem to efficiently adjust to publicly available information and finally, the strong form where only monopolistic access to information about prices does not seem to be a prevalent phenomenon in the investment community. Momentum, therefore, appears to violate the efficient market hypothesis in its weakest form (Ehsani and Linnainmaa (2022)). For the reason that past returns should not predict future returns if asset prices respond to new information immediately and to the right extent, unless past returns correlate with changes in systematic risk (Ehsani and Linnainmaa (2022)).

Initially, Levy (1967) introduces the term ârelative strengthâ as an earlier form of momentum. However, the momentum gained further attention after the influential work of Jegadeesh and Titman (1993) entitled âReturns to Buying Winners and Selling Losers: Implications for Stock Market Efficiencyâ. Jegadeesh and Titman (1993) noticed that stocks with higher past returns continue having a superior future return over the next three to twelve months, whereas stocks with lower past returns exhibit lower future returns over the next few months. They suggest that by buying past winners and selling past losers, investors can earn significant profits.

In the lecture *Asset Management: Investments*, bachelor students [1] of Banking and Finance voluntarily solve three Excel-based exercises called *Involving Activities* during the semester in order to gain exam points in advance. The individual exercises serve to deepen the material and apply the theory previously studied. The *Involving Activity 3* is designed to create a momentum strategy based on 18 given stocks. Thereby, the students build two momentum strategies by choosing their individual look-back period and holding period. The first strategy is a long-only strategy, referring to buying the stock with the anticipation of its increase in market value, and the second one a long-

[1] The course involves students and executive education participants to solve the relevant exercise but for simplicity reasons those two groups are summarized and only referred to as students from now on

short strategy, the latter further implies selling the stock with the anticipation of a decrease in market value. At the end of the *Involving Activity* they will compare their results with the buy-and-hold strategy.

Until now, the third *Involving Activity* could be influenced by the student which leads to several different approaches to solving it. This had the effect of every exercise having to be corrected manually resulting in a huge effort and time exposure. Thus, the following thesis aspires to update the *Involving Activity* among two different aspects. The exercise should, on one hand, remain to be individual for every student and, on the other hand, allow for an automated correction. Within the scope of this thesis, a correction tool has been developed and written in the programming language Python to evaluate the momentum strategies exercise of the students. As the primary requirement, the tool must be designed to allow for flexibility in the input data, lock-back and holding periods. The thesis is divided into three sections. The following section outlines an overview of influential literature on the cross-sectional and time-series momentum. Section 3 describes the *Involving Activity* and Python code in detail. Lastly, Section 4 outlines the conclusion and suggests an outlook for further implementations.

2 Literature Review

This section presents an overview of the main influential studies regarding the momentum strategy. Most literature focuses on the relative performance of securities, on the cross-section basis, thereby finding that securities that recently outperformed their peers over the past three to twelve months continue to outperform their peers on average over the next three to twelve months (Moskowitz et al. (2012)). Accordingly, the theoretical foundations of the cross-sectional momentum are outlined first. Afterwards, literature on the time-series momentum is summarized. Rather than focusing on the relative returns of securities in the cross-section, time-series momentum focuses exclusively on a securityâs own past return (Moskowitz et al. (2012)). Lastly, the two contrasting strategies are compared and an overview of all findings is given.

2.1 Cross-sectional Momentum

Jegadeesh and Titman (1993) documented that trading strategies which buy past winners and sell past losers realize significant abnormal returns over the 1965 to 1989 period. For example, the strategy which selects stocks based on their past six-month returns and holds them for six months realized a compounded excess return of 12.01% per year on average. Lewellen (2002) found that the momentum effect is significant for individual stocks and portfolios for seven to nine months after formation but quickly diminishes and turns to contrarian profits. This is in line with the findings of Jegadeesh and Titman (1993) and Moskowitz and Grinblatt (1999).

Rouwenhorst (1998) analysed return continuation in twelve European countries between 1980 and 1995 and found that a diversified portfolio of past winners outperformed a portfolio of past losers by approximately 1% per month. Rouwenhorst (1998) also validated the presence of momentum strategies in emerging markets.

Chan et al. (2000) investigated the profitability of momentum strategies for individual international equity stock indices by analysing past returns. The findings suggest the existence of significant momentum profits, particularly for short holding periods of less than four weeks. Griffin et al. (2005) expanded the research on 40 markets and found evidence for high profits in momentum strategy. They showed that momentum strategies generate substantial profits in a variety of markets. According to Kang et al. (2002), there were statistically significant abnormal profits associated with momentum strategies across eight distinct look-back and holding periods on the China stock market from 1993 to 2000. It was observed by Chan et al. (2019) that strategies focusing on past returns lead to significant profits over a six to twelve month timeframe, based on all the stocks listed on the NYSE, Amex, and Nasdaq. For instance, sorting stocks according to their prior six-month return generated a return spread of 8.8 percentage points during the subsequent six months. Chui et al. (2010) suggested that momentum strategies are found to be profitable in most regions around the world, both in developed and emerging markets, during the period 1980 to 2005. They also noted that the profitability

of momentum strategies varies across regions, with some regions exhibiting higher levels of profitability than others.

Naughton et al. (2008) examined momentum trading strategies for Shanghai Stock Exchange equities and found significant profits between 1995 to 2005, taking into account trading volume and utilizing different formation, and holding periods. Contrary, Hameed and Kusnadi (2002) reported limited evidence to support the presence of momentum in six emerging Asian stock markets over holding periods of three to twelve months. Hameed and Kusnadi (2002) found that these trading strategies consistently generate insignificant profits, indicating a lack of momentum effect in the Asian markets studied.

Jostova et al. (2019) documented strong evidence of momentum profitability in U.S. corporate bonds. Based on their dataset from 1991 to 2011, they revealed that past six-month winners outperform losers by an average of 59 basis points per month if held for six months. Additionally, momentum strategies were found to be profitable across formation and holding periods of three, nine, and twelve months. Beyhaghi and Ehsani (2016) also documented momentum profitability in loans of U.S. firms. Grinblatt et al. (1995) found that mutual fund momentum strategies were able to generate significant excess performance.

A concurrent study by Carhart (1997) discovered that purchasing mutual funds that were top-performers in the previous year while selling those that performed poorly can potentially result in higher returns. Miffre and Rallis (2007) found that momentum strategies applied to 31 US commodity futures contracts resulted in profitable returns over horizons that vary between one to twelve months in commodity futures markets.

Menkhoff et al. (2012) found significant excess returns of momentum strategies between past winning and losing currencies in the foreign exchange market. According to Bhojraj and Swaminathan (2006), analysing 38 country indices, winners outperform losers in the first three to twelve months after portfolio formation, but underperform losers in the following two years. Moreover, Okunev and White (2003) showed examining foreign exchange markets for Australia, Canada, France, Germany, Japan, Switzerland, the U.K., and the U.S. from January 1975 through June 2000 that the use of momentum strategy generated excess returns.

Asness et al. (2013) conducted a comprehensive analysis of eight distinct markets and asset classes which included individual stocks in the United States, the United Kingdom, continental Europe, and Japan, as well as country equity index futures, government bonds, currencies, and commodity futures. They revealed compelling evidence of momentum return premia across all the markets studied, indicating a pervasive and robust presence of momentum in the global financial markets.

2.2 Time-series Momentum

According to Moskowitz et al. (2012), time-series momentum is related to, but still different from, the momentum phenomenon in the finance literature which is primarily

cross-sectional in nature. The cross-sectional momentum approach involves selecting past winners and avoiding past losers, while time-series momentum focuses on trend-following based on a stock's past price movement. Predominantly, studies have mainly focused on futures markets and managed futures funds with regard to time-series momentum (Moskowitz et al. (2012), Hurst et al. (2017), Baltas and Kosowski (2013), Ham et al. (2019), Kim et al. (2016), Jin et al. (2019)).

As one of the first, Moskowitz et al. (2012) focused purely on a security's own past return and documented significant time-series momentum across very different asset classes and markets. Moreover, they found that the past twelve-month excess return of each instrument is a positive predictor of its future return. These findings are valid across a number of subsamples, look-back periods, and holding periods. Also, Moskowitz et al. (2012) showed that the existence and significance of time-series momentum is robust across horizons and asset classes, particularly when the look-back and holding periods are twelve months or less. In addition to Moskowitz et al. (2012), Baltas and Kosowski (2013) provided the first concrete empirical evidence of time series momentum using a broad daily data set of futures contracts. They showed that the time-series momentum strategy generates a statistically and economically significant mean return and alpha. Ham et al. (2019) documented significant time-series momentum profits in ten commodity futures from 2006 to 2018 in the Chinese market. Moreover, Hurst et al. (2017) reported that time-series momentum has been consistently profitable throughout the past 137 years. Specifically, they constructed combinations of one-month, three-month and twelve-month time-series momentum strategies from January 1880 to December 2016. Furthermore, they found that a trend-following strategy performed relatively similarly across a variety of economic environments and provided significant diversification benefits to a traditional allocation. Kim et al. (2016) re-examined the time-series momentum strategy using 55 liquid futures contracts. They found that time-series momentum only significantly outperforms a buy-and-hold strategy during the 1985 to 2009 period if the time-series momentum employed volatility scaling. Also, Huang et al. (2020), using the same data set as Moskowitz et al. (2012), found that the evidence on time-series momentum is weak and that time-series momentum across the global asset classes appears questionable.

Comparatively, less attention has been devoted to the most conventional of asset classes: common stocks. Lim et al. (2018) document strong time-series momentum effects in individual stocks in the US markets from 1927 to 2017. Moreover, time-series momentum is not specific to sub-periods, firm sizes, formation- and holding-period lengths, or geographic markets. Also, Georgopoulou and Wang (2017) documented a significant time-series momentum effect that is consistent and robust across global equity and commodity markets from 1969 to 2015. The findings of Shi and Zhou (2017) indicated that there is a time-series momentum effect in the short run and a contrarian effect in the long run in the Chinese stock market. The performances of the time series momentum and contrarian strategies are highly dependent on the look-back and holding periods and

firm-specific characteristics. Recently, Fang et al. (2021) show that profits may not be generally available for time-series momentum, considering 2.25 million monthly returns on over 20,000 US individual stocks from 1986 to 2017.

Later, other studies document inter-day time-series momentum effects. Gao et al. (2018) showed that the market return in the first half hour of the trading day predicts the market return in the last half hour. They showed an intra-day momentum pattern based on high frequency S&P 500 exchange-traded fund (ETF) data from 1993 to 2013. Jin et al. (2019) reported a similar intra-day time-series momentum across four Chinese Commodity future contracts from 2002 to 2013.

He and Li (2015) showed that the performance of momentum strategy is determined by both time horizon and the market dominance of momentum traders. Specifically, when the momentum traders dominate the market, the momentum strategy is profitable for a short time horizon and unprofitable when the time horizon is long.

DâSouza et al. (2016) stated the significant profitability of time-series momentum strategies in individual stocks in the US markets from 1927 to 2014 and in international markets since 1975. Moreover, the profitability of the strategy was robust for 16 different combinations of formation and holding periods, different benchmarks, and weighting systems in up and down markets. The outcomes of Zakamulin and Giner (2019) suggest that the long-only time-series momentum strategy is profitable, while the long-short one is not. In their study they included the monthly total returns on the Standard and Poorâs Composite stock price index, and the sample period begins in January 1857 and ends in December 2018.

2.3 Summary

In this subsection, the above-mentioned studies from the different markets are summarized and compared. Conclusive, there is strong evidence for momentum premium across different asset classes and markets, including individual stocks, futures, and commodities. The profitability of the momentum strategy is dependent on various factors such as time horizon, market dominance of momentum traders, and formation and holding periods. It can be distinguished between time-series momentum and cross-sectional momentum. Time-series momentum focuses on trend-following based on a stock's past price movement, while cross-sectional momentum involves buying past winners and selling past losers (Moskowitz et al. (2012)). Cheemaa et al. (2018) demonstrated that cross-sectional and time-series momentum returns in Japanese stock markets are significant. Moreover, they found evidence that time-series momentum returns exceed cross-sectional momentum returns. Due to its active position, which involves taking a net long or short position based on the direction of the market, while cross-sectional strategy is a zero-cost strategy that does not depend on the market state. Goyal and Jegadeesh (2017) adjusted for the net long positions of the time-series momentum strategy and found that, on average, the excess returns of cross-sectional and time-series momentum

strategies are generally equal.

3 Correction Tool

In the first section, the input data utilized is described and the Excel file serving as a template for the students to solve. Then a detailed explanation of the Python code used to correct the *Involving Activities* is given.

3.1 Data and Excel file

The purpose of the *Involving Activity* was to compare various risk and return measures of the student's long-only and long-short momentum strategies against a buy-and-hold strategy. The students were provided with monthly total return values of 18 stocks covering the time period from January 2002 to November 2022 (see Table 1). They were tasked with creating a cross-sectional momentum strategy based on a look-back and holding period which could range from three to twelve months as per the students' preference. These time periods are commonly studied and utilized in literature. Additionally, the students could choose to use either the arithmetic or geometric mean when calculating the return the ranking should be based on. The students were also expected to determine the number of shares to be bought or sold per month for both their long-only and long-short portfolios. Thus, the students would invest equally weighted (1/N) in the number of stocks according to their long-only or long-short strategy, respectively, and calculate the total return over their holding period.

Table 1: Given Data: Monthly total return values of the 18 stocks over a 20-year period
Extract of the monthly total return values of the 18 stocks that have been provided to the students. Note: for illustration purposes only five out of 18 stocks and four data points out of 20 years are presented.

Datum	ABB	CS	GEBERIT	GIVAUDAN	HOLCIM
01.01.2002	2762.73	1565.94	118.71	101.15	1670.80
01.02.2002	2978.23	1726.93	116.67	100.55	1730.38
01.03.2002	2762.73	1580.58	113.05	109.29	1757.79
01.04.2002	2336.04	1414.72	113.05	111.28	1770.90

3.2 Implementation

The Python code is designed to perform the thorough correction process of the *Involving Activity 3* for all students and executive education participants, substituting the manual correction in order to speed up the correction process. The only difference is that the executive education participants do not have a matriculation number, and the path to the submitted files has to be changed. For simplicity reason, only the correction tool for the students is described and analysed. The aforementioned individual choice of some parameters by the student posed a basic requirement to the correction tool.

Therefore, the Python code must be able to adapt flexibly to these input parameters in order to correct the respective *Involving Activity* of the student. Before diving into the detailed explanation of the code using flowcharts, a brief overview of the code structure will be provided by referring to the flowcharts to help understand the code better.

Flowcharts use various shapes to represent different elements. Each flowchart starts with a red rounded rectangle symbol representing the start of the process. Purple parallelograms are used to represent input or output, while orange rectangles describe the process being performed. Green diamonds indicate a decision point in the flowchart, while orange circles are connectors for `for-loops`. Lines are used to connect the various shapes and show their relationships and dependencies. Finally, a red rounded rectangle symbol is used to signify the end of the flowchart. If a process is further described in another flowchart the input or output values are stated directly next to it. The variables used in the flowcharts are identical to the ones in the code.

3.2.1 Overview

The flowchart in Figure 1 a simple outline of the Python code.

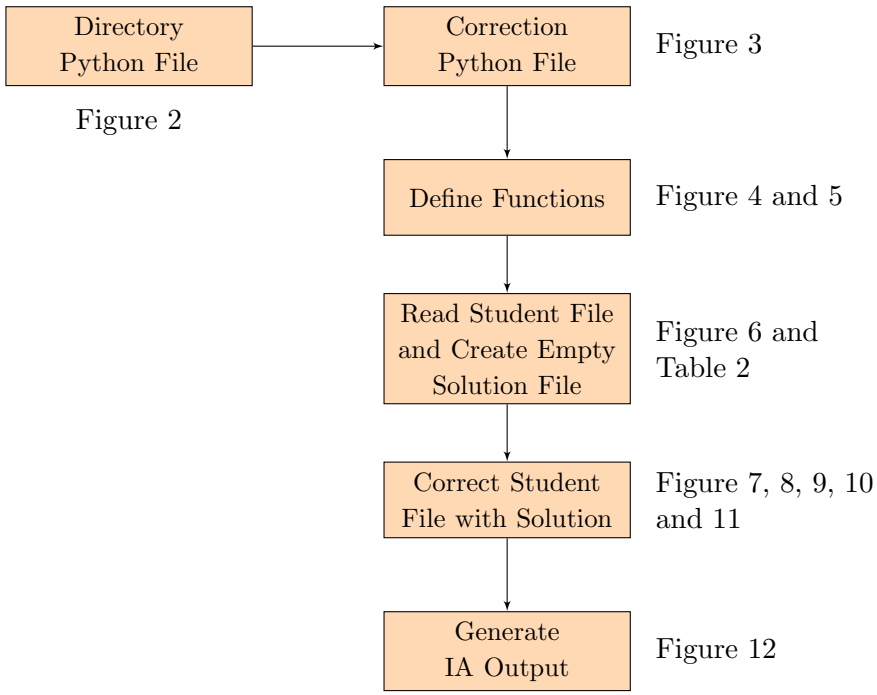

Figure 1: Flowchart Overview

The code structure is divided into two python files. The first file involves the setup of the directory to access the submitted student files. This process is further described in Section 3.2.2. The second file, outlined in Section 3.2.3, introduces the correction process for each student and the *IA Output*. In the correction file three functions are defined (see Section 3.2.4). First, the function `load_workbook_range` is defined (see Figure 4). Every worksheet that will be loaded into a dataframe in python has the same formation.

Therefore, with this function data with a specified range in a worksheet can be loaded and a dataframe is returned. It can also optionally include headers and an index column with a specified name. The function `nthRoot` is really straight forward and will not be explained in detail using a flowchart. After that, the function `correction(filenames)`, as shown in Figure 5 is defined. The `correction(filenames)` consists of several steps. First, the student file and the empty solution file is loaded into dataframes (see Section 3.2.5). Figure 6 illustrates how the student file is loaded into dataframes. This step also involves saving important input parameters and definding variables. Table 2 shows the data used to calculate the solution of the monthly return. All other calculations base on the student's value for taking consequential errors into account. After that, the student file will be corrected with the solution (see Section 3.2.6). For that, the solution is calculated and the difference in those dataframes is built for then assign points for each exercise. In Section 3.2.7, the *IA Output* is generated which contains information of the student, the points he achieved and if he passed or failed the exercise. This information can then be used to perform the mass evaluation on OLAT. Also the Flowchart 12 illustretes that process in detail.

3.2.2 Directory Python File

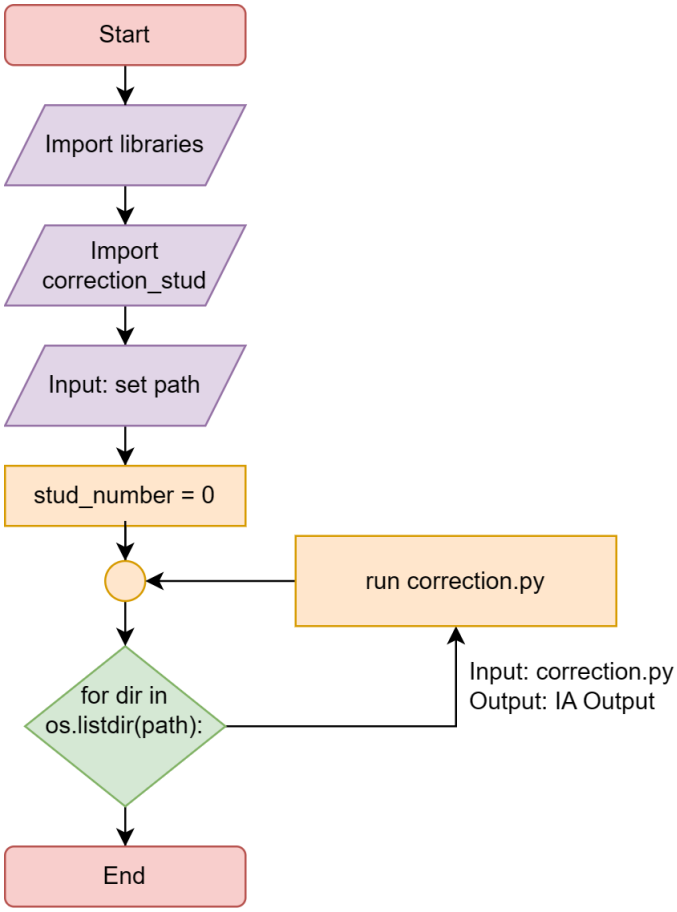

Figure 2: Flowchart Directory.py

The main part of the Directory Flowchart (see Figure 2) is the `for loop` iterating through each student file. To enable this `for loop`, the `correction_stud.py` (see Figure 3) needs to be imported which contains the second Python file with the correction and *IA Output* in it. Then the code defines a file path, where the student's submitted files are stored, and initializes a counter for the number of students processed, `stud_number`, to zero. Next, the code uses a `for loop` that iterates through each directory in the specified path using the `os.listdir` function. The `os.listdir(path)` function retrieves a list of all the directories in the specified path. The variable `dir` will hold the name of each directory in the list in each iteration of the `loop`. With this `for loop`, access is granted to the submission folder which contains the submitted Excel sheets of the students. In the code, this is represented with the filenames. Therefore, in each iteration of the loop, the code accesses the *Involving Activity* of one of the student's directories within the file path. This allows the code to perform the correction process for each student's *Involving Activity* individually, instead of processing all the submissions in a single run. For each directory, the code uses the `glob` library to search for Excel files with a ".xlsx" extension. The `os.path.join` function is used to construct the full path to each student's submissions directory. The `if` statement checks if the filenames list is empty or not. If filenames is empty, meaning that no Excel files were found in the current directory, then the `correction_stud.correction` function is not called, and the code continues to the next directory in the `os.listdir` iteration. The code keeps track of the number of students processed by increasing the `stud_number` variable each time it processes a directory. The code then calls the `correction_stud.correction` function with the first Excel file in the filenames list as an argument. This function is defined in the `correction_stud.py` and performs the correction on the *Involving Activities*. This function is described in detail in the following paragraph (see Figure 3).

3.2.3 Correction Python File

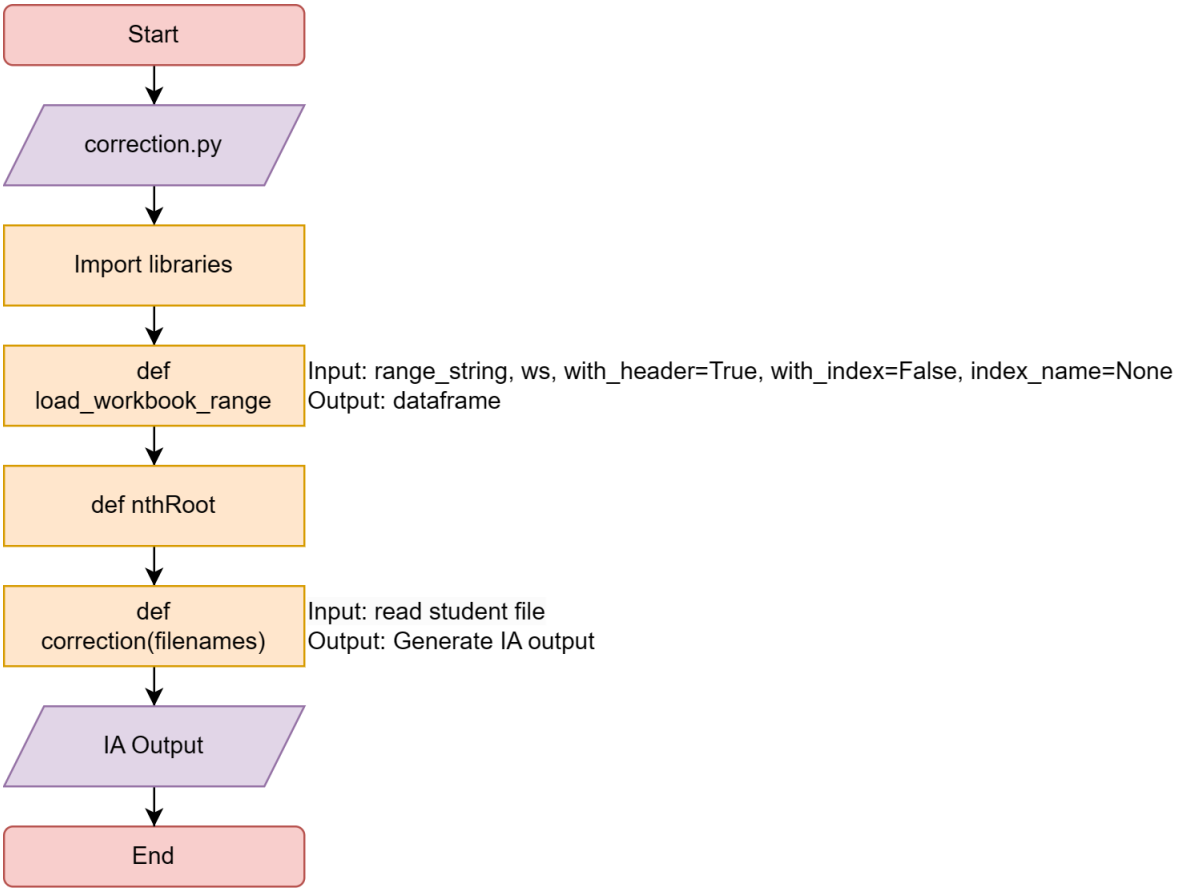

Figure 3: Flowchart correction.py

The correction process within the code involves the use of dataframes for efficient processing of data. Given the 18 shares and a 20-year time period, dataframes provide a convenient and scalable method for organizing and analyzing the data in Python. First, the code defines the function `load_workbook_range` that can be appreciated in Figure 4. This function is used to load specific data ranges from an Excel workbook. Additionally, the function `nthRoot` is used to find the "n^{th}" root of a given value "x". The core correction process is performed by the function `correction(filenames)` which is described in detail in Figure 5.

3.2.4 Define Functions

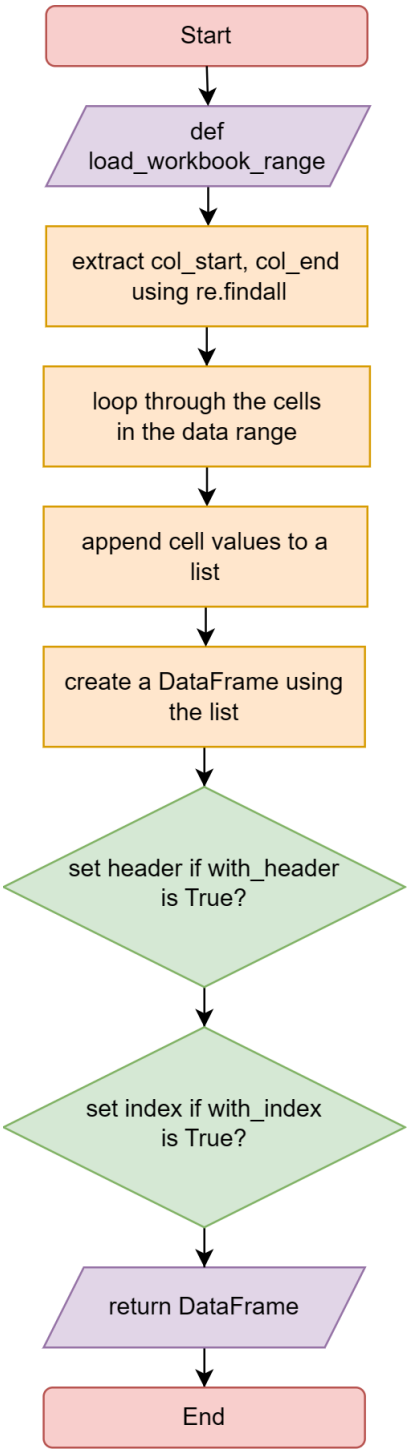

Figure 4: Flowchart load_workbook_range()

The `load_workbook_range(range_string, ws, with_header=True, with_index=False, index_name=None)` function is a function that loads data from an Excel sheet into a pandas dataframe. It takes the following parameters:

- `range_string`: a string specifying the range of cells in the worksheet to be loaded

into the dataframe. The format of the string should be in the form of "A1:Z100"

- `ws`: a worksheet object representing the worksheet in the workbook.
- `with_header`: a boolean value indicating whether the first row of the data should be used as the header for the dataframe.
- `with_index`: a boolean value indicating whether the data should be set as the index of the dataframe.
- `index_name`: a string specifying the name of the index if `with_index` is set to True. If `with_index` is False, this parameter is ignored.

The function first uses the `re` module to extract the start and end columns of the data range. It then loops through the cells in the data range and appends the cell values as a list to `data_rows`. Next, a dataframe is created from the `data_rows` and the columns are set to the column names obtained from the `range_string` using the `get_column_interval` function. If `with_header` is True, the first row of the dataframe is set as the header. If the `with_index` flag is True and the `index_name` is not None, the dataframe is given an index using the `index_name` as the label. The `print` statement then prints the columns of the dataframe. Finally, the dataframe is returned.

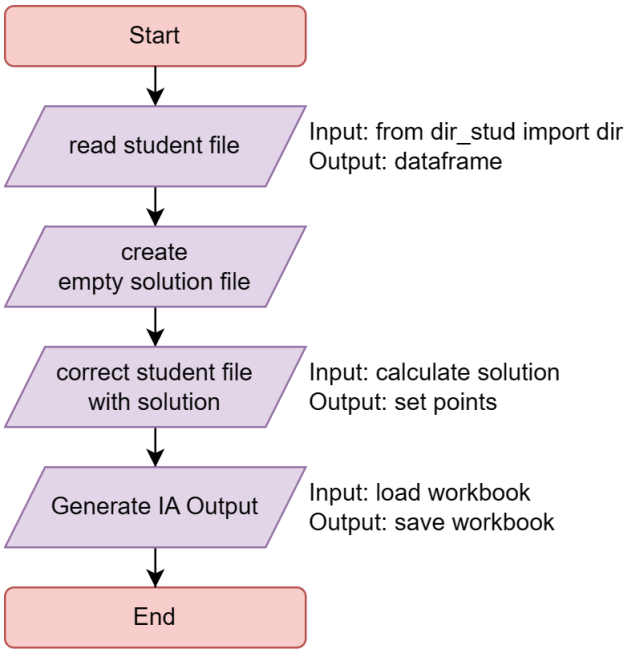

Figure 5: Flowchart def correction(filenames)

The function `correction(filenames)` as shown in Figure 5 outlines the correction process for each student's *Involving Activity*. The process consists of several steps each of them described in further detail in the respective figures.

3.2.5 Read Student File and Create Empty Solution File

The first step involves reading the student's file illustrated in Figure 6. This step consists of loading the data from the student's Excel workbook into a dataframe for processing. Next, an empty solution file is created. This file will be used to store the solution. Once the student file and the solution file have been prepared, the code proceeds to compare the results of the student with the solution ones, as described in the following figures. At the end of the Python file, the *IA Output* is generated, as presented in Figure 12.

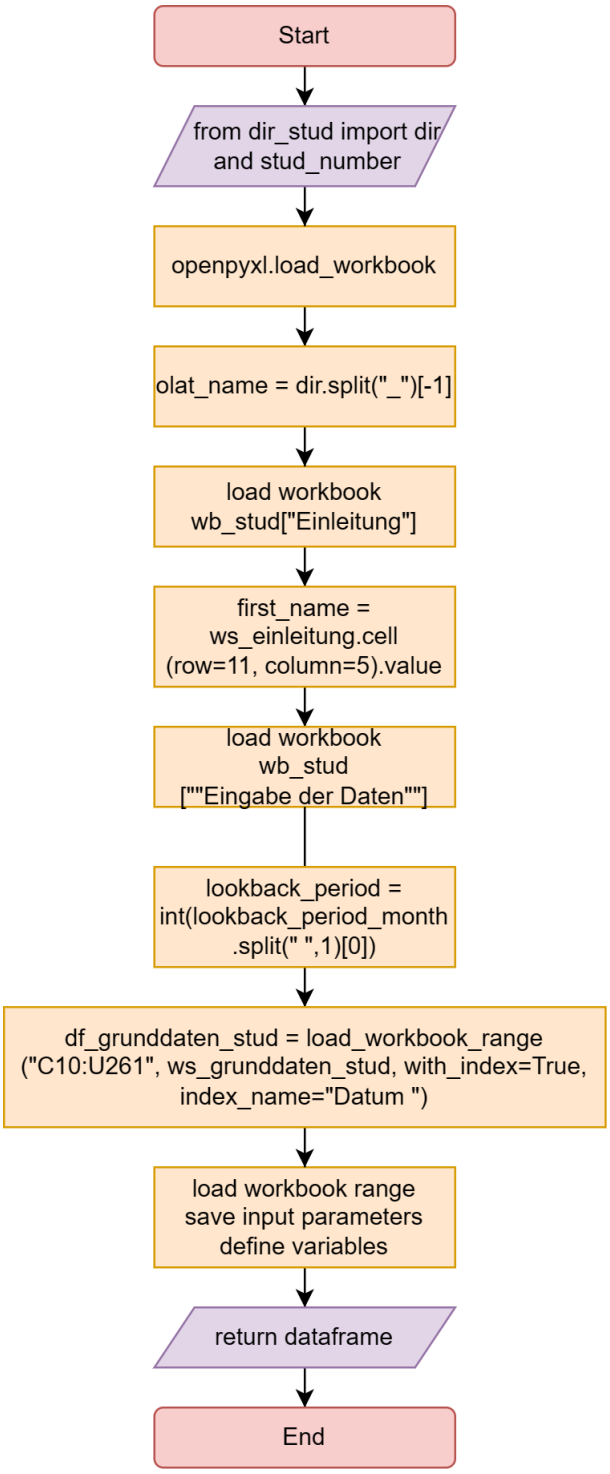

Figure 6: Flowchart read student file

This code, as shown in Figure 6, firstly imports two variables, `dir` and `stud_number` from the `dir_stud` module. The "from" keyword is used to specify the module from which the variables are being imported. After this line has been executed, the `stud_number` and `dir` variables are accessible in the current scope and can be used just like any other variables.

Next, the code uses the `openpyxl` library to load an Excel file into memory and create a workbook object. The `load_workbook` function is called with the filenames argument and the `data_only` argument is set to True, which means that any formulas in the workbook will be replaced with their calculated values. The resulting workbook is stored in the `wb_stud` variable which can be used to access and manipulate the data in the workbook.

The code then splits the value stored in the dir variable into separate parts using the `split` method which separates a string into a list of substrings based on a specified delimiter. The delimiter defined is: "_". The resulting list of substrings is then indexed using square bracket notation with -1 as the index. The -1 index retrieves the last element in the list. This value, after being split by "_", is assigned to the `olat_name` variable.

This line of code uses the openpyxl library to load an Excel file into memory and create a workbook object. The `load_workbook` function is called with the filenames argument which is assumed to be a list of filenames (specifically, in this code, it is a list containing only one filename). After loading the workbook, the code accesses a specific worksheet in the workbook, called "Einleitung". This worksheet is loaded into the `ws_einleitung` variable. Then, the code searches for a specific cell in the worksheet, at row 11 and column 5, using the `.cell()` method. The value of this cell is retrieved using the value attribute and stored in the `first_name` variable. It is important to note that in the `.cell()` method, the first row and first column in the worksheet are both numbered 1, not 0. Thus, row 11 and column 5 refer to the 11th row and 5th column in the worksheet, respectively. The process will be repeated for other relevant variables. After that, the maximal reachable points in each exercise are saved as a variable.

Then, the next worksheet "Eingabe der Daten" is loaded into the `ws_eingabe_der_daten`. The code again accesses a specific cell in this worksheet where the students could choose their look-back period. The code takes the value stored in the `lookback_period_month` variable and splits it into separate parts using the split method. The delimiter used here is a space character: " ". The resulting list of substrings is then indexed using square bracket notation with 0 as the index. The 0 index retrieves the first element in the list. The value of the first element in the list, after being split by a space character, is then converted to an integer using the `int` function and assigned to the `lookback_period` variable. For example, if the value of `lookback_period_month` is "12 months", then after splitting the value using split(" ", 1), the list would be ["12", "months"]. The value of "12" would be converted to an integer using int("12"), which would result in the integer value 12. This integer value would then be assigned to the lookback_period variable. The same process goes with the holding period. Some more variables are defined which are not going to be further explained in this section.

The line, `ws_grunddaten_stud = wb_stud["Grunddaten"]` accesses the worksheet named `"Grunddaten"` in the workbook object stored in the `wb_stud` variable. The result of this expression is a worksheet object, which is assigned to the `ws_grunddaten_stud` variable.

The line, `df_grunddaten_stud = load_workbook_range("C10:U261", ws_grunddaten_stud, with_index=True, index_name="Datum ")` uses the `load_workbook_range` function (see Figure 4) to load data from a specified range of cells in the worksheet into a dataframe. The first argument to the `load_workbook_range function`, `"C10:U261"`, is a string specifying the range of cells to load using the A1 notation. This string specifies that the data should be loaded from the cells in columns C to U and rows 10 to 261. The second argument to the `load_workbook_range` function, `ws_grunddaten_stud`, is the worksheet object from which the data should be taken. The `with_index` argument is set to `True` implying that the first row of the specified range of cells will be used as the index (row labels) of the resulting dataframe. The `index_name` argument is set to `"Datum "` which specifies the name to use for the index (row labels) of the resulting dataframe. The result of the `load_workbook_range` function call is assigned to the `df_grunddaten_stud` variable, which will now contain a pandas dataframe with the loaded data. The remaining worksheets will be loaded into a dataframe the same way.

After loading the studentile into dataframes in Python the empty solution file with the same workbook range is loaded into Python. The time period in the empty solution file in the sheet "Grunddaten" had to be adjusted by starting the time period already at first December 2001 instead of first January 2002 (see Table 2). Because of this adjustment the .pct() function could be used to calculate the monthly return starting in January 2002. Otherwise the first monthly return would be stated in February 2002. This was done to ensure that the use of the percentage change function would result in a comparable index between the solution values and the student values.

Table 2: Shifted Data: Monthly total return values of the 18 stocks over a 20-year period

Extract of the monthly total return values of the 18 stocks that already start in December 2001 instead of January 2002. Note: for illustration purposes only five out of 18 stocks and four data points out of 20 years are presented.

Datum	ABB	CS	GEBERIT	GIVAUDAN	HOLCIM
01.12.2001	2762.73	1565.94	118.71	101.15	1670.80
01.01.2002	2978.23	1726.93	116.67	100.55	1730.38
01.02.2002	2762.73	1580.58	113.05	109.29	1757.79
01.03.2002	2336.04	1414.72	113.05	111.28	1770.90

3.2.6 Correct Student File with Solution

Having organized all the necessary data in Python, the file is prepared to start iterating through each worksheet and using it to correct and grade the corresponding student worksheet.

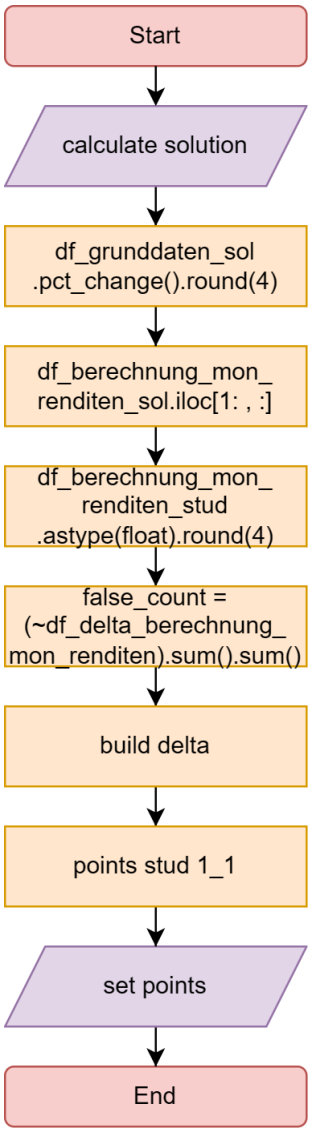

Figure 7: Flowchart Sheet Monatliche Rendite

In the first worksheet, shown in Figure 7, the students had to calculate the monthly return of every share. Firstly, the solution based on the given data is calculated. The `.pct_change().round(4)` function calculates the percentage change of each value in the `df_grunddaten_sol` dataframe and stores the result in a new dataframe `df_berechnung_mon_renditen_sol`. The result is rounded to four decimals. To adjust the solution data, the `.iloc()` function is used to remove the first row from the `df_berechnung_mon_renditen_sol` dataframe. The `.iloc()` method is utilized to access specific rows and columns in a dataframe based on their integer position. The `[1: , :]` syntax specifies that all rows starting from the second row (index 1) and all columns `(:)` should be selected. Thus, this line of code selects all rows in `df_berechnung_mon_renditen_sol` starting from the second row, and all columns. The function `.astype(float).round(4)` converts the data in the `df_berechnung_mon_renditen_stud` dataframe to the float data type and rounds the result to four decimal places. This step is necessary as the result calculation differs

between Excel and Python.

Next, the difference between the solution and student dataframes is calculated by equating the student dataframe with the solution dataframe. To determine any disparities, a delta dataframe is created which indicates instances where the difference between the two dataframes was non-zero, thereby expressing a discrepancy between the student's result and the expected solution. Contrarily, instances with no difference are recorded as True. Then, the amount of False values in the `df_delta_berechnung_mon_renditen` dataframe get counted. The operator inverts the Boolean values in the dataframe so True values become False and vice versa. The sum method is then used to count the number of False values in each column of the dataframe, and the sum method is called again to find the total number of False values across all columns. This total is stored in the `false_count` variable. After calculating how many values are false in the student dataframe, the points for this first exercise that the students receive are calculated. The `points_stud_1_1` is the maximum of two values. Therefore, the value of `points_stud_1_1` is reduced based on the number of False values in the `df_delta_berechnung_mon_renditen` dataframe. Every correct value gives a sub-point. The sub-points add up to the maximum number of points which could be achieved in the exercise. So, for every false value, a sub-point is subtracted and saved the achieved points in the variable `points_stud_1_1`.

The sequence of rounding the values and building the delta, counting the false values, and setting the points for the exercise is the same process in every sheet. Therefore, this process will not again be described in detail in the following extensions. Then a new dataframe is created by adding one to the monthly return dataframe. This will later be useful for calculating the geometric mean.

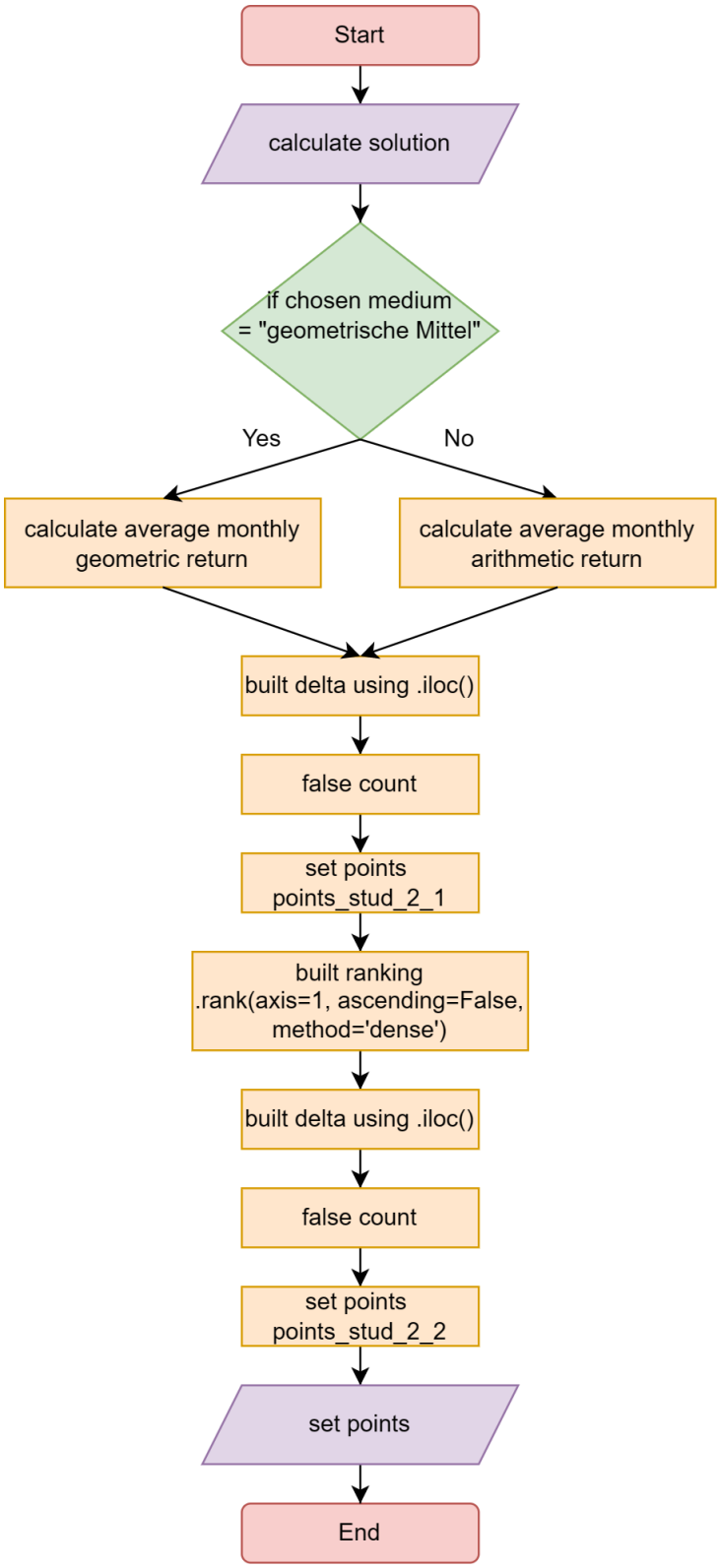

Figure 8: Flowchart Sheet Ranking

As mentioned before, the students could choose if they rather wanted to base the ranking on the arithmetic or geometric mean. The if statement, shown in Figure 8

above, checks whether the chosen medium is equal to the string "geometrische Mittel".

If this is deemed true, the average monthly return of the 18 stocks based on the student's individual look-back period is calculated. For that the `.rolling()` function is applied which uses the plus one return data (see Figure 7). This method provides rolling windows over the data. The size of the window is passed as a parameter in the function `.rolling(window)` and is set to the look-back period. The `.apply()` method is used in conjunction with the `.rolling()` function to apply the `np.prod` function to each rolling window of the dataframe. The `.apply()` method applies the `np.prod` function to each rolling window, which calculates the product of all the values within the window. The `raw=True` parameter indicates that the function is applied to the underlying numpy array of the dataframe rather than to each column independently. The result is then passed to the `nthRoot` function which calculates the "n^{th}" root of the product where n is equal to look-back period. Finally, -1 is subtracted from the result since the returns were initially added by one. This creates a new dataframe that represents the geometric mean.

If `mittel_ranking` is not equal to "geometrische Mittel", the code inside the else block is executed. This is the case when the student has chosen the arithmetic mean. The mean function is then used to calculate the mean of all values in the window for each row. This creates a new dataframe that represents the arithmetic mean of `df_berechnung_mon_renditen_stud` over a rolling window of the size `lookback_period`.

Note that all calculations, except for the monthly return, are based on the student's values in order to include and reward consequential errors also with points.

After generating the solution, the delta is calculated by rounding both the student and solution dataframes to four decimal points. However, the rolling function can cause empty rows with irrelevant information, such as "look-back period," the `.iloc` function is used to delete these rows from both dataframes before calculating the delta.

As a result of deleting some rows in the dataframes, the number of points that will be subtracted for each false value needs to be modified accordingly. This adjustment ensures that the score accurately reflects the student's performance in the task even though some rows were deleted during the analysis. The number of false values in the delta is used to calculate the points for exercise 2.1.

Based on the total return, the ranking by applying the rank function to `df_ranking_stud_lb` is built. The rank function assigns a rank to each element in a dataframe based on the values of the elements. The argument axis is set to 1, meaning that the ranking will be done along the columns (across the row axis). The argument `ascending` is set to False, meaning that the ranking will be done in descending order. The argument method is set to `dense`, meaning that the ranks will be assigned with no gaps, i.e. if two elements have the same value they will receive the same rank and the next rank will be incremented by 2. The resulting dataframe `df_ranking_sol_rank` will have the same number of rows as `df_ranking_stud_lb` and the same number of columns but each value in the dataframe will be replaced with its rank. By building the delta, the same amount of rows are

deleted simultaneously, and the false values which serve as the basis for determining the points for exercise 2.2 are counted.

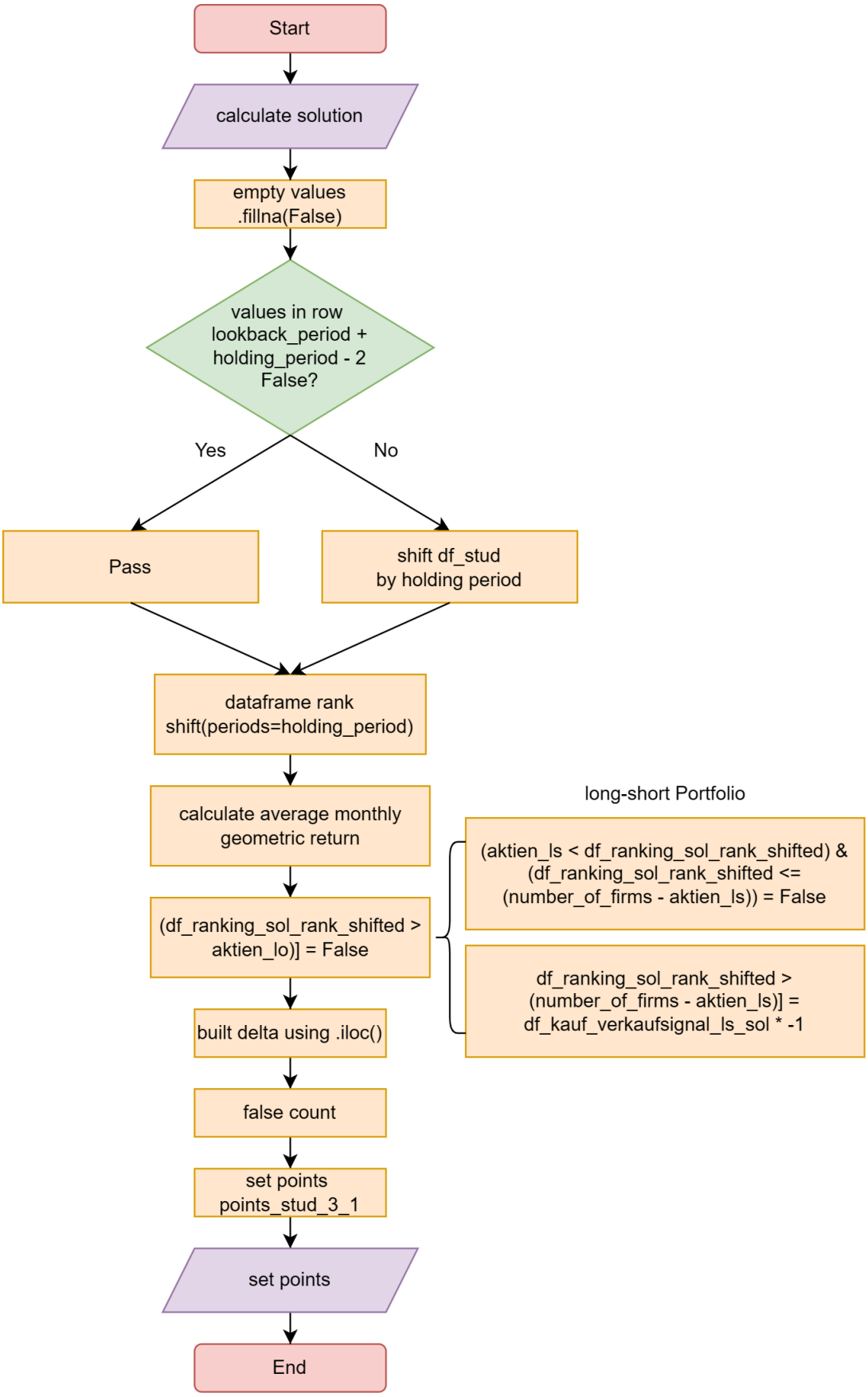

Figure 9: Flowchart Sheet Kauf- & Verkaufsignal

In Figure 9, displayed above, the students invested in the number of stocks according to their long-only or long-short strategy and calculate the total return over their holding period. Some students performed a backward test, meaning they calculated the total return at the beginning of the holding period instead of at the end. To determine whether a backward test was conducted or not, the `.fillna(False)` function was used to replace any missing values with False. Then the code checks whether all the values in the row `lookback_period + holding_period - 2` of `df_kauf_verkaufsignal_lo_stud` are False or not. In case there is at least one True value in the row `lookback_period + holding_period - 1` it is definite that no backward test was performed. If there was no True value, the pass statement was used as a placeholder. If a backward test was performed, the code proceeds to the else statement. In that instance, in order to compare the results later, the values in the student's dataframe were shifted (using the shift method of the pandas dataframe) by the holding period.

Then the `.iloc()` function is used again to delete the rows that aren't necessary for the correction. Afterwards, the dataframe with the ranks in it is shifted by the amount of holding periods so that the correct ranks will get compared with the number of stocks in the long-only portfolio for calculating the rolling return. Again by using the `.rolling()` function the moving average monthly geometric return will be calculated with the window set to the holding period.

For the long-only portfolio, the students were instructed to invest equally weighted (1/N) in the number of stocks according to their long-only strategy. Therefore, the number of stocks the student has chosen for the long-only portfolio is compared with the rank dataframe, and when the rank was greater than the number of stocks, it was replaced with False. After that, empty rows were deleted from both the solution and student dataframes using the `.iloc()` function, and all values were rounded to four decimal points. The delta was calculated again to compare the solution and student dataframes, and the number of false values in the delta was used to set the points for exercise 3.1.

The process is the same for the long-short portfolio except that the students buy and sell the number of shares according to their long-short strategy. Therefore, the code filters out any firms for which the rank is higher than the number of stocks in the long-short portfolio, but not higher than the total number of firms minus the number of stocks in the long-short portfolio. This is the range of ranks that would be included in the long-short portfolio. Any stocks outside this range are set to False in the `df_kauf_verkaufsignal_ls_sol` dataframe indicating they should not be included in the portfolio. Because the worst-ranked stocks are sold according to the long-short portfolio the returns thereof must be multiplied by -1. This computation reflects the act of selling them in the long-short portfolio.

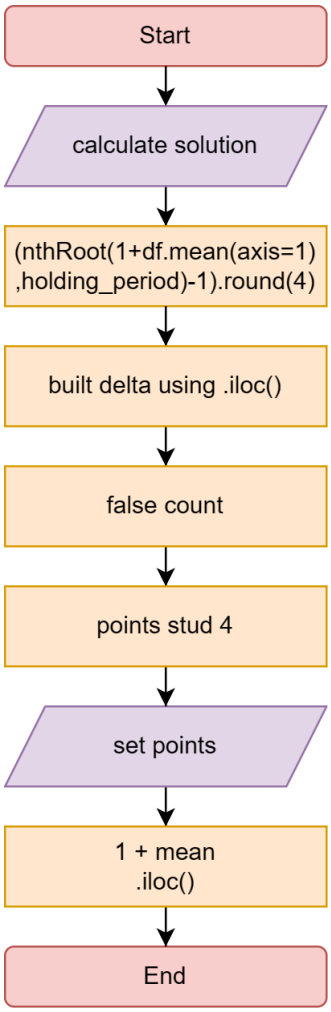

Figure 10: Flowchart Sheet Monatliche Portfoliorenditen

In the presented Figure 10, the monthly portfolio returns were calculated for three different strategies, namely long-only, short-long, and buy-and-hold. The `df_kauf_verkaufsignal_lo_stud.me` calculates the mean of each row (axis=1) in the dataframe and returns a new dataframe with a single column. The calculation for the long-only and long-short portfolio then takes the "n^{th}" root of `(1 + mean returns)` with n set to the holding period and subtracts 1. The final result is rounded to four decimal places using the round function. The result is stored in a new column in the `df_monatliche_portfoliorenditen_so` dataframe. For the buy-and-hold strategy, the `.mean()` function was applied to calculate the monthly returns. After rounding the values and deleting the unnecessary rows, the deltas were computed, and the points for exercise 4 were set. The monthly return of each strategy was increased by one and the empty rows were removed again using the `.iloc()` function.

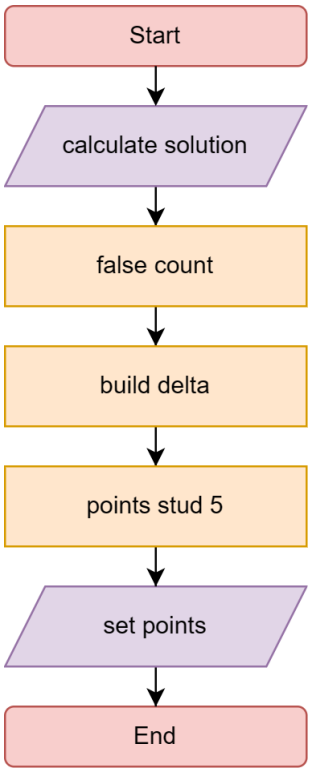

Figure 11: Flowchart Sheet Gesamtrendite und SR

In this sheet, shown in Figure 11 above, different risk and return measurements are calculated. The monthly arithmetic average return is computed using the `.mean()` function for each of the three investment strategies. Because the calculation should base on the student's value, the annualized arithmetic return is obtained using the `.iloc()` function to select a specific column from the dataframe `df_gesamtrendite_sr_stud`. To calculate the geometric return, a new dataframe is created by adding one to the monthly portfolio return of the student for each strategy. The `.iloc()` function is used to select the relevant data, and the monthly geometric average return was computed using the `nthRoot` function and the `np.prod` method. The annualized geometric return is computed using the `.iloc()` function again to select a specific column from the dataframe.

The monthly variance is calculated using the `np.var` function with the `ddof` parameter set to 1. This parameter sets the divisor used in the calculation (`N - ddof`) where N represents the number of elements. By setting the `ddof` to 1, the variance is computed in the same way as the students use the function VAR.S in Excel. The yearly variance is computed using the `.iloc()` function to select the relevant data.

The monthly and yearly volatility is obtained using the `nthRoot` function and the `.iloc()` function to select the specific row of the data. The Sharpe ratio is computed using the standard formula and plotted in the data.

All the risk and return measurements are combined into a new dataframe with an index of 1 to 9 to enable the comparison with the student's dataframe. The dataframes are

rounded to four decimal points, and the delta is computed by subtracting the student's dataframe from the solution. The number of false values in the delta is counted, and the points for this exercise are set to 5.

3.2.7 Generate IA Output

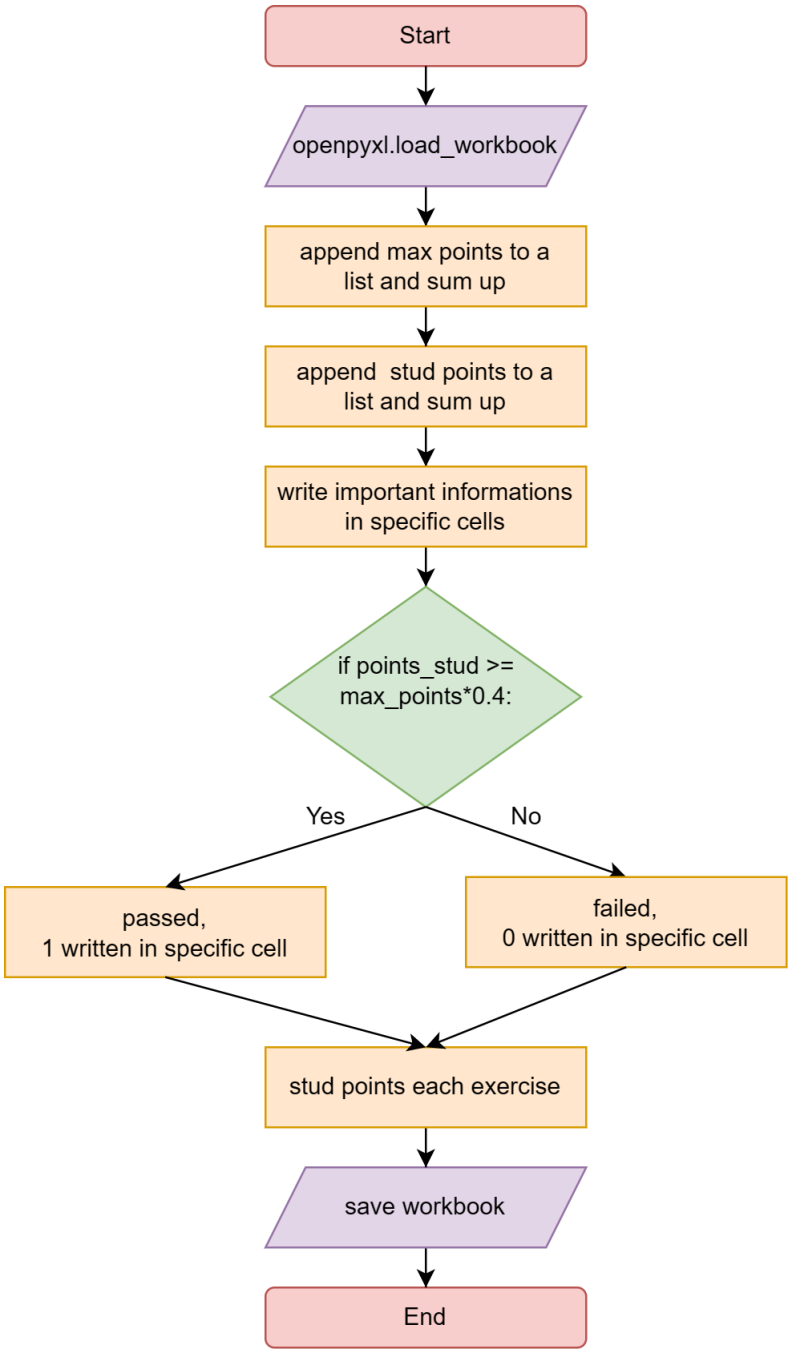

Figure 12: Flowchart Sheet IA Output

After calculating and correcting the necessary dataframes, the student information and points are loaded into an empty Excel file named "IA Output". The maximum achievable

points for each exercise are stored in a list, and the total number of achievable points is calculated by taking the sum of this list. The student's achieved points are calculated in the same manner. Next, the relevant student information such as first and last name, number of achieved points, and OLAT name are written into the worksheet by selecting the appropriate cell and defining the row and column in which the data should be noted. The row of the cell is determined by additionally adding the `stud_number` variable which keeps track of the current student being processed and prevents overwriting the previous data in the Excel worksheet. To determine whether the student passed or failed the *Involving Activity*, an if-else statement is used. If the student passed the exercise, a 1 is written in the corresponding cell of the workbook. Otherwise, a 0 is noted. This simplifies the process of uploading the results to OLAT. In addition, the code records the number of points the student achieved in each individual exercise. Finally, the workbook is saved, and the correction process is repeated for the next student's file.

3.3 Correction Manual

In this subsection, a manual for the *Headcoach*, who is responsible for correcting the *Involving Activity*, is provided. Firstly, in order to run the correction, it is necessary to install Python and a code editor such as Visual Studio Code. Once installed, both Python files can be opened.

It is important to note that all the required files need to be downloaded from OLAT and saved on the computer. The path in the `dir_stud` Python file must be adjusted accordingly. Additionally, in the `correction_stud` file, the path for the loaded solution workbook and *IA Output* also needs to be adjusted. Please note that for the solution workbook, the data needs to be shifted by one month, starting one week earlier, as shown in Table 2. Therefore, besides the students' files, the empty solution file with the shifted data and the empty *IA Output* file must be stored on the computer in the specified path.

If the *Headcoach* only needs to update the return dates and replace a share with another, no further adjustments are required to run the correction.

In the event of a change in the layout of the "Einleitung" sheet, it is important to note that the location of data within the sheet may be affected. As a result, it may be necessary to adjust the "row=" and "column=" arguments while saving the following variables:

- `first_name`
- `last_name`
- `matriculation_number`

This will allow the code to access and manipulate the relevant data in the updated sheet layout.

Similarly, in the event of any changes made to the layout of the "Eingabe der Daten" sheet, it may be necessary to adjust the "row=" and "column=" arguments while saving the following variables:

- `lookback_period_month`
- `holding_period_month`
- `mittel_ranking`
- `aktien_lo`
- `aktien_ls`

When a bigger or smaller time period is used, or more or less shares are considered, it may be necessary to make the following adjustments:

- Update input variable `number_of_firms` and `time_period`
- Update `load_workbook_range()` for every worksheet

If there is a change in the points awarded per exercise or the riskfree rate changes, these variables can be easily adjusted in the code to reflect the new values.

Additionaly, if the sheets in the *Involving Activity* workbook are renamed, it is essential to rename them accordingly in the code as well.

For generating the *IA Output* and in the case that there has been an update to the layout of the *IA Output* Excel file, the corresponding arguments "row=" and "column=" in the code must be modified to match the new layout:

- `matriculation_number`
- `first_name`
- `last_name`
- `points_stud`
- `olat_name`
- `'1' or '0'`
- `points_stud_1_1, points_stud_2_1, points_stud_2_2, points_stud_3_1, points_stud_3_2, points_stud_4, points_stud_5`

The passing limit, which is currently set to 40%, can be adjusted by increasing or decreasing this number as desired.

After running the code to generate the *IA Output*, the *Headcoach* can open the file and copy the columns "OLAT" and "Bestanden." These columns can then be used to perform a mass evaluation of the students' results on OLAT.

4 Conclusion and further implementations

The main objective of this thesis was to develop an automated correction tool to efficiently correct the *Involving Activity 3* in the course *Asset Management: Investments*. To ensure flexibility with regard to lock-back and holding periods, an option for individual selection has been incorporated into the *Involving Activity* at the outset of the exercise. Additionally, the input data comprising historical stock prices of 18 shares has been loaded into Python. As a result, the correction tool designed in Python is capable of accommodating updates to input data, such as changes in share prices or dates, and also on the look-back and holding period. Moreover, in order to provide the students with more flexibility in determining their rankings, an option was implemented where students are allowed to choose between using either the arithmetic or geometric mean as the basis for their ranking. This would allow students to select the method that best aligns with their personal preferences and understanding of the data. The correction tool is designed to consider consequential errors by using the student's submitted values for correction except for the first exercise for calculating monthly returns. The tool has successfully automated the correction of the *Involving Activity*, obviating the need for manual correction. The *Headcoach* is only required to make certain modifications in the code, such as adjusting the filepath. The tool provides the capability to adjust the loaded workbook range for each sheet or the saved input variables (e.g. the number of firms) in the event that modifications are required.

Nevertheless, some potential limitations must be considered. Firstly, because of loading the range of every workbook into a dataframe in Python, the range is predetermined. One potential solution to this limitation is to automate the process of adjusting the range. This could be achieved by implementing a function that can read the active range, as a number of firms and time period, in Excel based on certain criteria, such as colour-coding or specific cell values. Additionally, it may be possible to optimize the code to handle larger datasets more efficiently, thereby reducing the need for manual adjustments. However, it is important to note that these solutions may require additional development and testing to ensure their effectiveness and reliability.

Secondly, to address the drawback of not being able to provide feedback to students, a possible extension of the tool could include generating personalised feedback for each student, such as a sheet with a point overview or their Excel sheet color-coded that highlights correct and incorrect answers. This would enhance the usefulness of the tool by providing students with valuable feedback on their performance, enabling them to identify areas for improvement and promoting a more engaged and effective learning experience.

However, the use of the correction tool remains an effective and efficient approach for correcting the *Involving Activity 3*. The objective and automated nature of the correction process enables faster processing times and helps to minimize errors that may occur when correcting by hand, making it a valuable addition to the course *Asset*

Management: Investments.

Bibliography

Asness, C. S., Moskowitz, T. J., and Pedersen, L. H. (2013). Value and momentum everywhere. *The Journal of Finance*, 68(3):929–985.

Baltas, A.-N. and Kosowski, R. (2013). Momentum strategies in futures markets and trend-following funds. *Available at SSRN 1968996*.

Beyhaghi, M. and Ehsani, S. (2016). The cross-section of expected returns in the secondary corporate loan market. *The Review of Asset Pricing Studies*, 7(2):243–277.

Bhojraj, S. and Swaminathan, B. (2006). Macromomentum: Returns predictability in international equity indices. *The Journal of Business*, 79(1):429–451.

Carhart, M. M. (1997). On persistence in mutual fund performance. *The Journal of Finance*, 52(1):57–82.

Chan, K., Hameed, A., and Tong, W. (2000). Profitability of momentum strategies in the international equity markets. *The Journal of Financial and Quantitative Analysis*, 35(2):153–172.

Chan, L. K., Jegadeesh, N., and Lakonishok, J. (2019). The profitability of momentum strategies. *Financial Analysts Journal*, 55(6):80–90.

Cheemaa, M. A., Narteab, G. V., and Szulczyk, K. R. (2018). Cross-sectional and time-series momentum returns and market dynamics: evidence from japan. *Applied Economics*, 50(23):2600–2612.

Chui, A. C., Titman, S., and Wei, K. J. (2010). Individualism and momentum around the world. *The Review of Financial Studies*, 65(1):361–392.

DâSouza, I., Srichanachaichok, V., Wang, G., and Yao, Y. (2016). The enduring effect of time-series momentum on stock returns over nearly 100-years. *Available at SSRN 2720600*.

Ehsani, S. and Linnainmaa, J. A. (2022). Factor momentum and the momentum factor. *The Journal of Finance*, 77(3):1877–1919.

Fama, E. F. (1970). Efficient capital markets: A review of theory and empirical work. *Journal of Finance*, 25(2):383–417.

Fang, J., Hao, W., and Wongchoti, U. (2021). Time-series momentum in individual stocks: is it there and where to look? *Applied Economics*, 54(18):2048–2066.

Gao, L., Han, Y., Li, S. Z., and Zhou, G. (2018). Market intraday momentum. *Journal of Financial Economics*, 129(2):394–414.

Georgopoulou, A. and Wang, J. G. (2017). The trend is your friend: Time- series momentum strategies across equity and commodity markets. *Review of Finance*, 21(4):1557–1592.

Goyal, A. and Jegadeesh, N. (2017). Cross-sectional and time-series tests of return predictability: What is the difference? *The Review of Financial Studies*, 31(5):1784–1824.

Griffin, J. M., Ji, X., and Martin, J. S. (2005). Global momentum strategies. *The Journal of Portfolio Management*, 31(2):23–39.

Grinblatt, M., Titman, S., and Wermers, R. (1995). Momentum investment strategies, portfolio performance, and herding: A study of mutual fund behavior. *The American Economic Review*, 85(5):1088–1105.

Ham, H., Cho, H., Kim, H., and Ryu, D. (2019). Timeâseries momentum in chinaâs commodity futures market. *Journal of Futures Markets*, 39(12):1515–1528.

Hameed, A. and Kusnadi, Y. (2002). Momentum strategies: Evidence from pacific basin stock markets. *The Journal of Financial Research*, 25(3):383–397.

He, X.-Z. and Li, K. (2015). Profitability of time series momentum. *Journal of Banking and Finance*, 53:140–157.

Huang, D., Li, J., Wang, L., and Zhou, G. (2020). Time series momentum: Is it there? *Journal of Financial Economics*, 135(3):774–794.

Hurst, B., Ooi, Y. H., and Pedersen, L. H. (2017). A century of evidence on trend-following investing. *The Journal of Portfolio Management*, 44(1):15–29.

Jegadeesh, N. and Titman, S. (1993). Returns to buying winners and selling losers: Implications for stock market efficiency. *The Journal of Finance*, 48(1):65–91.

Jin, M., Kearney, F., Li, Y., and Yang, Y. C. (2019). Intraday timeâseries momentum: Evidence from china. *Journal of Investment Management*, 40(4):632–650.

Jostova, G., Nikolova, S., Philipov, A., and Stahel, C. W. (2019). Momentum in corporate bond returns. *The Review of Financial Studies*, 26(7):1649–1693.

Kang, J., Liu, M.-H., and Xiaoyan Ni, S. (2002). Contrarian and momentum strategies in the china stock market: 1993â2000. *Pacific-Basin Finance Journal*, 10(3):243–265.

Kim, A. Y., Tse, Y., and Wald, J. K. (2016). Time series momentum and volatility scaling. *Journal of Financial Markets*, 30:103–124.

Levy, R. A. (1967). Relative strength as a criterion for investment selection. *The Journal of Finance*, 22(4):595–610.

Lewellen, J. (2002). Momentum and autocorrelation in stock returns. *The Review of Financial Studies*, 15(2):533–564.

Lim, B. Y., Wang, J. G., and Yao, Y. (2018). Time-series momentum in nearly 100 years of stock returns. *Journal of Banking and Finance*, 97:283–296.

Menkhoff, L., Sarno, L., Schmeling, M., and Schrimpf, A. (2012). Currency momentum strategies. *Journal of Financial Economics*, 106(3):660–684.

Miffre, J. and Rallis, G. (2007). Momentum strategies in commodity futures markets. *Journal of Banking and Finance*, 31(6):1863–1886.

Moskowitz, T. J. and Grinblatt, M. (1999). Do industries explain momentum? *The Review of Financial Studies*, 54(4):1249–1290.

Moskowitz, T. J., Ooi, Y. H., and Pedersen, L. H. (2012). Time series momentum. *Journal of Finance*, 104(2):228–250.

Naughton, T., Truong, C., and Veeraraghavan, M. (2008). Momentum strategies and stock returns: Chinese evidence. *Pacific-Basin Finance Journal*, 16(4):476–492.

Okunev, J. and White, D. (2003). Do momentum-based strategies still work in foreign currency markets? *Journal of Financial and Quantitative Analysis*, 38(2):425–447.

Rouwenhorst, K. G. (1998). International momentum strategies. *The Journal of Finance*, 53(1):267–284.

Shi, H.-L. and Zhou, W.-X. (2017). Time series momentum: Is it there? *Physica A*, 483:309–318.

Zakamulin, V. and Giner, J. (2019). Trend following with momentum versus moving averages: a tale of differences. *Quantitative Finance*, 20(6):985–1007.

Appendices

A Code

A.1 Directory Student

```
import os
import glob


import correction_stud


path = 'C:\\Users\\senns\\Documents\\Uni_Stuff\\2022\\Bachelor
↪  arbeit\\Final_Take\\files_for_correction_stud\\
ita_IA_3_2022-12-23T20-12-29_674'
stud_number = 0
for dir in os.listdir(path):
    filenames = glob.glob(os.path.join(path + '\\' + dir +
    ↪  '\\2_submissions\\' , "*.xlsx"))
    stud_number +=1
    if filenames != []:
        correction_stud.correction(filenames[0])
```

A.2 Correction Student

```
from ast import Return
import math
from fileinput import close
from importlib.machinery import FrozenImporter
from importlib.resources import path
from multiprocessing.sharedctypes import Value
import os
from queue import Empty
from sqlite3 import Row
from tkinter import E
from tkinter.messagebox import YES
from tkinter.tix import COLUMN
import pandas as pd
import numpy as np

import openpyxl     #library
from statistics import variance
from statistics import stdev
from scipy.stats import gmean
```

```
import numpy as np
import os
import xlwings as xw


from openpyxl import load_workbook
from openpyxl.utils import get_column_interval
import re


def load_workbook_range(range_string, ws, with_header=True,
    with_index=False, index_name=None):
    col_start, col_end = re.findall("[A-Z]+", range_string)

    data_rows = []
    for row in ws[range_string]:
        data_rows.append([cell.value for cell in row])

    df = pd.DataFrame(data_rows, columns=get_column_interval(col_start,
        col_end))

    if (with_header):
        df.columns = df.iloc[0]
        df = df.iloc[1:]

    if (with_index and index_name is not None):
        df = df.set_index(index_name, drop=True)
        print(df.columns)

    return df

def nthRoot(x, n):
        return x**(1/float(n))

def correction(filenames):
    # Read Student File
    from dir_stud import dir
    from dir_stud import stud_number
    wb_stud = openpyxl.load_workbook(filenames, data_only=True) #read
        excel file
    olat_name = dir.split("_")[-1]
    ws_einleitung = wb_stud["Einleitung"]
    first_name = ws_einleitung.cell(row=11, column=5).value
    last_name = ws_einleitung.cell(row=13, column=5).value
    matrikelnummer = ws_einleitung.cell(row=15, column=5).value
```

```
62
63      points_1_1 = 6
64      points_2_1 = 6
65      points_2_2 = 6
66      points_3_1 = 6
67      points_3_2 = 6
68      points_4 = 6
69      points_5 = 15
70
71      ws_eingabe_der_daten = wb_stud["Eingabe der Daten"]
72      lookback_period_month = ws_eingabe_der_daten.cell(row=11,
        ↪  column=12).value
73      lookback_period = int(lookback_period_month.split(" ",1)[0])
74      holding_period_month = ws_eingabe_der_daten.cell(row=13,
        ↪  column=12).value
75      holding_period = int(holding_period_month.split(" ",1)[0])
76      mittel_ranking = ws_eingabe_der_daten.cell(row=22, column=11).value
77      aktien_lo = ws_eingabe_der_daten.cell(row=26, column=12).value
78      aktien_ls = ws_eingabe_der_daten.cell(row=28, column=12).value
79      number_of_firms = 18
80      riskfree = 0
81      time_period = 250
82
83      ws_grunddaten_stud = wb_stud["Grunddaten"]
84      df_grunddaten_stud = load_workbook_range("C10:U261",
        ↪  ws_grunddaten_stud, with_index=True, index_name="Datum ")
85
86      ws_berechnung_mon_renditen_stud = wb_stud["Berechnung mon. Renditen"]
87      df_berechnung_mon_renditen_stud = load_workbook_range("C13:U263",
        ↪  ws_berechnung_mon_renditen_stud, with_index=True,
        ↪  index_name="Datum ")
88
89      ws_ranking_stud = wb_stud["Ranking"]
90      df_ranking_stud_lb = load_workbook_range("C13:U263", ws_ranking_stud,
        ↪  with_index=True, index_name="Datum ")
91      df_ranking_stud_rank = load_workbook_range("C267:U517",
        ↪  ws_ranking_stud, with_index=True, index_name="Datum ")
92
93      ws_kauf_verkaufsignal_stud = wb_stud["Kauf- & Verkaufsignal"]
94      df_kauf_verkaufsignal_lo_stud = load_workbook_range("C14:U264",
        ↪  ws_kauf_verkaufsignal_stud, with_index=True, index_name="Datum ")
95      df_kauf_verkaufsignal_ls_stud = load_workbook_range("W14:AO264",
        ↪  ws_kauf_verkaufsignal_stud, with_index=True, index_name="Datum ")
96
```

```
97      ws_monatliche_portfoliorenditen_stud = wb_stud["Monatliche
        ↪  Portfoliorenditen"]
98      df_monatliche_portfoliorenditen_stud = load_workbook_range("C13:F263",
        ↪  ws_monatliche_portfoliorenditen_stud, with_index=True,
        ↪  index_name="Datum ")
99
100     ws_gesamtrendite_sr_stud = wb_stud["Gesamtrendite & SR"]
101     df_gesamtrendite_sr_stud = load_workbook_range("D11:F20",
        ↪  ws_gesamtrendite_sr_stud)
102
103
104     # Create Solution File
105     wb_sol =
        ↪  openpyxl.load_workbook('C:\\Users\\senns\\Documents\\Uni_Stuff\\
106     2022\\Bachelorarbeit\\Final_Take\\IA_3_HS22.xlsx', data_only=True)
107     ws_grunddaten_sol = wb_sol["Grunddaten"]
108     df_grunddaten_sol = load_workbook_range("C10:U261", ws_grunddaten_sol,
        ↪  with_index=True, index_name="Datum ")
109
110     ws_berechnung_mon_renditen_sol = wb_sol["Berechnung mon. Renditen"]
111     df_berechnung_mon_renditen_sol = load_workbook_range("C13:U263",
        ↪  ws_berechnung_mon_renditen_sol, with_index=True, index_name="Datum
        ↪  ")
112
113     ws_ranking_sol = wb_sol["Ranking"]
114     df_ranking_sol_lb = load_workbook_range("C13:U264", ws_ranking_sol,
        ↪  with_index=True, index_name="Datum ")
115     df_ranking_sol_rank = load_workbook_range("C267:U517", ws_ranking_sol,
        ↪  with_index=True, index_name="Datum ")
116
117     ws_kauf_verkaufsignal_sol = wb_sol["Kauf- & Verkaufsignal"]
118     df_kauf_verkaufsignal_lo_sol = load_workbook_range("C14:U264",
        ↪  ws_kauf_verkaufsignal_sol, with_index=True, index_name="Datum ")
119     df_kauf_verkaufsignal_ls_sol = load_workbook_range("W14:AO264",
        ↪  ws_kauf_verkaufsignal_sol, with_index=True, index_name="Datum ")
120
121     ws_monatliche_portfoliorenditen_sol = wb_sol["Monatliche
        ↪  Portfoliorenditen"]
122     df_monatliche_portfoliorenditen_sol = load_workbook_range("C13:F263",
        ↪  ws_monatliche_portfoliorenditen_sol, with_index=True,
        ↪  index_name="Datum ")
123     ws_gesamtrendite_sr_sol = wb_sol["Gesamtrendite & SR"]
124     df_gesamtrendite_sr_sol = load_workbook_range("C11:F20",
        ↪  ws_gesamtrendite_sr_sol, with_index=True)
125
```

```
126
127
128     #Sheet monatliche Rendite
129     df_berechnung_mon_renditen_sol =
        ↪  df_grunddaten_sol.pct_change().round(4)
130     df_berechnung_mon_renditen_sol =
        ↪  df_berechnung_mon_renditen_sol.iloc[1: , :]
131     df_berechnung_mon_renditen_stud =
        ↪  df_berechnung_mon_renditen_stud.astype(float).round(4)
132     df_delta_berechnung_mon_renditen = df_berechnung_mon_renditen_stud ==
        ↪  df_berechnung_mon_renditen_sol
133     false_count = (~df_delta_berechnung_mon_renditen).sum().sum()
134     points_stud_1_1 = max(points_1_1 -
        ↪  points_1_1/number_of_firms/time_period*false_count,0)
135
136     df_berechnung_mon_renditen_add_one_sol =
        ↪  df_berechnung_mon_renditen_sol + 1
137     df_berechnung_mon_renditen_stud_add_one =
        ↪  df_berechnung_mon_renditen_stud + 1
138
139     #Sheet Ranking obere tabelle Rendite gemÃŒss lookback period
140     if mittel_ranking == "geometrische Mittel":
141         df_ranking_sol_lb =
            ↪  nthRoot((df_berechnung_mon_renditen_stud_add_one).rolling(
142         window=lookback_period).apply(np.prod, raw=True),lookback_period)
            ↪  - 1
143     else:
144         df_ranking_sol_lb =
            ↪  df_berechnung_mon_renditen_stud.rolling(window=lookback_period)
145         .mean()
146     df_delta_ranking_lb =
        ↪  df_ranking_sol_lb.iloc[lookback_period-1:].round(4) ==
        ↪  df_ranking_stud_lb.iloc[lookback_period-1:].astype(float).round(4)
147     false_count = (~df_delta_ranking_lb).sum().sum()
148     points_stud_2_1 =
        ↪  max(points_2_1-(points_2_1/((number_of_firms*time_period)-
149     ((lookback_period-1)*number_of_firms)))*false_count,0)
150
151     #Sheet Ranking untere Tabelle Ranking bilden
152     df_ranking_sol_rank = df_ranking_stud_lb.rank(axis=1, ascending=False,
        ↪  method='dense')
153     df_delta_ranking_rank = df_ranking_sol_rank.iloc[lookback_period-1:]
        ↪  == df_ranking_stud_rank.iloc[lookback_period-1:]
154     false_count = (~df_delta_ranking_rank).sum().sum()
```

```
155     points_stud_2_2 =
        ↪  max(points_2_2-(points_2_2/((number_of_firms*time_period)-
156     ((lookback_period-1)*number_of_firms)))*false_count,0)
157
158     #Kauf- & Verkaufsignal lo, long-only Portfolio
159     df_kauf_verkaufsignal_lo_stud =
        ↪  df_kauf_verkaufsignal_lo_stud.fillna(False)
160     if df_kauf_verkaufsignal_lo_stud.iloc[lookback_period+holding_period-
161     2].sum() == False: #when stud made a backward test
162         pass
163     else: df_kauf_verkaufsignal_lo_stud =
        ↪  df_kauf_verkaufsignal_lo_stud.shift(periods=holding_period)
164     df_berechnung_mon_renditen_add_one_sol =
        ↪  df_berechnung_mon_renditen_stud_add_one.iloc[lookback_period-1:]
165     df_kauf_verkaufsignal_lo_sol =
        ↪  df_kauf_verkaufsignal_lo_sol.iloc[lookback_period-1:]
166     df_ranking_sol_rank_shifted =
        ↪  df_ranking_stud_rank.shift(periods=holding_period) #compare
        ↪  correct rank for calculating rolling return
167     df_kauf_verkaufsignal_lo_sol =
        ↪  (df_berechnung_mon_renditen_add_one_sol).rolling(window=
168     holding_period).apply(np.prod, raw=True) - 1
169     df_kauf_verkaufsignal_lo_sol[(df_ranking_sol_rank_shifted >
        ↪  aktien_lo)] = False
170     df_kauf_verkaufsignal_lo_sol =
        ↪  df_kauf_verkaufsignal_lo_sol.iloc[holding_period:].astype(float).round(4)
171     df_kauf_verkaufsignal_lo_stud =
        ↪  df_kauf_verkaufsignal_lo_stud.iloc[holding_period+lookback_period
172     -1:].astype(float).round(4)
173     df_delta_kauf_verkaufsignal_lo = df_kauf_verkaufsignal_lo_sol ==
        ↪  df_kauf_verkaufsignal_lo_stud
174     false_count = (~df_delta_kauf_verkaufsignal_lo).sum().sum()
175     points_stud_3_1 =
        ↪  max(points_3_1-(points_3_1/((number_of_firms*time_period)-((holding_period+lookback_
176
177     #Kauf- & Verkaufsignal ls, long-short Portfolio
178     df_kauf_verkaufsignal_ls_stud =
        ↪  df_kauf_verkaufsignal_ls_stud.fillna(False)
179     if
        ↪  df_kauf_verkaufsignal_ls_stud.iloc[lookback_period+holding_period-2].sum()
        ↪  == False: #when stud made a backward test
180         pass
181     else:
182         df_kauf_verkaufsignal_ls_stud =
            ↪  df_kauf_verkaufsignal_ls_stud.shift(periods=holding_period)
```

```
183         df_monatliche_portfoliorenditen_stud =
            ↪  df_monatliche_portfoliorenditen_stud.shift(periods=holding_period)
184     df_kauf_verkaufsignal_ls_sol =
        ↪  df_kauf_verkaufsignal_ls_sol.iloc[lookback_period-1:]
185     df_kauf_verkaufsignal_ls_sol =
        ↪  df_berechnung_mon_renditen_add_one_sol.rolling(window=holding_period).apply(np.prod,
        ↪  raw=True) - 1
186     df_kauf_verkaufsignal_ls_sol[(aktien_ls < df_ranking_sol_rank_shifted)
        ↪  & (df_ranking_sol_rank_shifted <= (number_of_firms - aktien_ls))]
        ↪  = False
187     df_kauf_verkaufsignal_ls_sol[df_ranking_sol_rank_shifted >
        ↪  (number_of_firms - aktien_ls)] = df_kauf_verkaufsignal_ls_sol * -1
188     df_kauf_verkaufsignal_ls_sol =
        ↪  df_kauf_verkaufsignal_ls_sol.iloc[holding_period:].astype(float).round(4)
189     df_kauf_verkaufsignal_ls_stud =
        ↪  df_kauf_verkaufsignal_ls_stud.iloc[holding_period+lookback_period-1:].astype(float).
190     df_delta_kauf_verkaufsignal_ls = df_kauf_verkaufsignal_ls_sol ==
        ↪  df_kauf_verkaufsignal_ls_stud
191     false_count = (~df_delta_kauf_verkaufsignal_ls).sum().sum()
192     points_stud_3_2 =
        ↪  max(points_3_2-(points_3_2/((number_of_firms*time_period)-((holding_period+lookback_
193
194     #Monatliche Portfoliorenditen
195     df_monatliche_portfoliorenditen_sol["Long-only"] =
        ↪  (nthRoot(1+df_kauf_verkaufsignal_lo_stud.mean(axis=1),holding_period)-1).round(4)
196     df_monatliche_portfoliorenditen_sol["Long-Short"] =
        ↪  (nthRoot(1+df_kauf_verkaufsignal_ls_stud.mean(axis=1),holding_period)-1).round(4)
197     df_monatliche_portfoliorenditen_sol["Buy and Hold"] =
        ↪  (df_berechnung_mon_renditen_stud.mean(axis=1)).round(4)
198     df_monatliche_portfoliorenditen_stud =
        ↪  df_monatliche_portfoliorenditen_stud.astype(float)
199     df_delta_monatliche_portfoliorenditen =
        ↪  df_monatliche_portfoliorenditen_sol.iloc[holding_period+lookback_period-1:]
        ↪  ==
        ↪  df_monatliche_portfoliorenditen_stud.iloc[holding_period+lookback_period-1:].astype(
200     false_count = (~df_delta_monatliche_portfoliorenditen).sum().sum()
201     points_stud_4 =
        ↪  max(points_4-(points_4/((3*number_of_firms)-((holding_period+lookback_period-1)*3)))
202
203
204     df_monatliche_portfoliorenditen_sol["Long-only 1+r"] =
        ↪  df_monatliche_portfoliorenditen_sol["Long-only"]+1
205     df_monatliche_portfoliorenditen_sol["Long-Short 1+r"] =
        ↪  df_monatliche_portfoliorenditen_sol["Long-Short"]+1
```

```
206     df_monatliche_portfoliorenditen_sol["Buy and Hold 1+r"] =
        ↪  df_monatliche_portfoliorenditen_sol["Buy and Hold"]+1
207
208     df_monatliche_portfoliorenditen_sol =
        ↪  df_monatliche_portfoliorenditen_sol.iloc[lookback_period+holding_period-1:]
209
210     #Gesamtrendite und SR
211     monatl_arithm_durchschnittsrendite_sol_lo =
        ↪  df_monatliche_portfoliorenditen_stud["Long-only"].iloc[lookback_period+holding_perio
212     monatl_arithm_durchschnittsrendite_sol_ls =
        ↪  df_monatliche_portfoliorenditen_stud["Long-Short"].iloc[lookback_period+holding_peri
213     monatl_arithm_durchschnittsrendite_sol_bah =
        ↪  df_monatliche_portfoliorenditen_stud["Buy and
        ↪  Hold"].iloc[lookback_period+holding_period-1:].mean()
214     annualisierte_arithm_rendite_sol_lo =
        ↪  (1+df_gesamtrendite_sr_stud.iloc[0,0])**12-1
215     annualisierte_arithm_rendite_sol_ls =
        ↪  (1+df_gesamtrendite_sr_stud.iloc[0,1])**12-1
216     annualisierte_arithm_rendite_sol_bah =
        ↪  (1+df_gesamtrendite_sr_stud.iloc[0,2])**12-1
217     df_gmean_lo =
        ↪  1+df_monatliche_portfoliorenditen_stud["Long-only"].iloc[lookback_period+holding_per
218     df_gmean_ls =
        ↪  1+df_monatliche_portfoliorenditen_stud["Long-Short"].iloc[lookback_period+holding_pe
219     df_gmean_bah = 1+df_monatliche_portfoliorenditen_stud["Buy and
        ↪  Hold"].iloc[lookback_period+holding_period-1:]
220     monatl_geom_durchschnittsrendite_sol_lo =
        ↪  nthRoot(np.prod(df_gmean_lo),time_period-lookback_period-holding_period-1)-1
221     monatl_geom_durchschnittsrendite_sol_ls =
        ↪  nthRoot(np.prod(df_gmean_ls),time_period-lookback_period-holding_period-1)-1
222     monatl_geom_durchschnittsrendite_sol_bah =
        ↪  nthRoot(np.prod(df_gmean_bah),time_period-lookback_period-holding_period-1)-1
223     annualisierte_geom_rendite_sol_lo =
        ↪  (1+df_gesamtrendite_sr_stud.iloc[2,0])**12-1
224     annualisierte_geom_rendite_sol_ls =
        ↪  (1+df_gesamtrendite_sr_stud.iloc[2,1])**12-1
225     annualisierte_geom_rendite_sol_bah =
        ↪  (1+df_gesamtrendite_sr_stud.iloc[2,2])**12-1
226     var_monatlich_sol_lo =
        ↪  np.var(df_monatliche_portfoliorenditen_stud["Long-only"], ddof=1)
227     var_monatlich_sol_ls =
        ↪  np.var(df_monatliche_portfoliorenditen_stud["Long-Short"], ddof=1)
228     var_monatlich_sol_bah =
        ↪  np.var(df_monatliche_portfoliorenditen_stud["Buy and Hold"],
        ↪  ddof=1)
```

```
    var_jaehrlich_sol_lo = df_gesamtrendite_sr_stud.iloc[5,0]*12
    var_jaehrlich_sol_ls = df_gesamtrendite_sr_stud.iloc[5,1]*12
    var_jaehrlich_sol_bah = df_gesamtrendite_sr_stud.iloc[5,2]*12
    vola_monatlich_sol_lo = nthRoot(df_gesamtrendite_sr_stud.iloc[4,0],2)
    vola_monatlich_sol_ls = nthRoot(df_gesamtrendite_sr_stud.iloc[4,1],2)
    vola_monatlich_sol_bah = nthRoot(df_gesamtrendite_sr_stud.iloc[4,2],2)
    vola_jaehrlich_sol_lo = nthRoot(df_gesamtrendite_sr_stud.iloc[6,0],2)
    vola_jaehrlich_sol_ls = nthRoot(df_gesamtrendite_sr_stud.iloc[6,1],2)
    vola_jaehrlich_sol_bah = nthRoot(df_gesamtrendite_sr_stud.iloc[6,2],2)
    sharpe_ratio_sol_ls = (annualisierte_arithm_rendite_sol_lo -
    ↪  riskfree)/ vola_jaehrlich_sol_lo
    sharpe_ratio_sol_lo = (annualisierte_arithm_rendite_sol_ls -
    ↪  riskfree)/ vola_jaehrlich_sol_ls
    sharpe_ratio_sol_bah = (annualisierte_arithm_rendite_sol_bah -
    ↪  riskfree)/ vola_jaehrlich_sol_bah

    gesamtrendite_sr_sol = {'Long-only':
    ↪  [monatl_arithm_durchschnittsrendite_sol_lo,annualisierte_arithm_rendite_sol_lo,monat
    ↪  'Long-Short':
    ↪  [monatl_arithm_durchschnittsrendite_sol_ls,annualisierte_arithm_rendite_sol_ls,monat
    ↪  'Buy and Hold':
    ↪  [monatl_arithm_durchschnittsrendite_sol_bah,annualisierte_arithm_rendite_sol_bah,mon
    df_gesamtrendite_sr_sol = pd.DataFrame(data=gesamtrendite_sr_sol,
    ↪  index=[1,2,3,4,5,6,7,8,9]) #set index like this to compare stud
    ↪  and solution
    df_gesamtrendite_sr_stud =
    ↪  df_gesamtrendite_sr_stud.astype(float).round(4)
    df_gesamtrendite_sr_sol = df_gesamtrendite_sr_sol.round(4)
    df_delta_gesamtrendite_sr = df_gesamtrendite_sr_sol ==
    ↪  df_gesamtrendite_sr_stud
    false_count = (~df_delta_gesamtrendite_sr).sum().sum()
    points_stud_5 = max(points_5 - points_5/27*false_count,0) #bc 27
    ↪  measurements

    #Generate IA Output
    wb_IA_output =
    ↪  openpyxl.load_workbook('C:\\Users\\senns\\Documents\\Uni_Stuff\\2022\\Bachelorarbeit
    ↪  data_only=True)
    ws_IA_output = wb_IA_output["IA Output"]

    list_max_points =
    ↪  [points_1_1,points_2_1,points_2_2,points_3_1,points_3_2,points_4,points_5]
    max_points = sum(list_max_points)
    list_points_stud =
    ↪  [points_stud_1_1,points_stud_2_1,points_stud_2_2,points_stud_3_1,points_stud_3_2,poi
```

```
257     points_stud = sum(list_points_stud)
258     ws_IA_output.cell(row=6+stud_number, column=3).value = matrikelnummer
259     ws_IA_output.cell(row=6+stud_number, column=4).value = first_name
260     ws_IA_output.cell(row=6+stud_number, column=5).value = last_name
261     ws_IA_output.cell(row=6+stud_number, column=6).value = points_stud
262     ws_IA_output.cell(row=6+stud_number, column=7).value = olat_name
263     if points_stud >= max_points*0.4:
264         ws_IA_output.cell(row=6+stud_number, column=8).value = '1' #means
            ↪  passed
265     else: ws_IA_output.cell(row=6+stud_number, column=8).value = '0'
        ↪  #means failed
266     ws_IA_output.cell(row=6+stud_number, column=10).value =
        ↪  points_stud_1_1
267     ws_IA_output.cell(row=6+stud_number, column=11).value =
        ↪  points_stud_2_1
268     ws_IA_output.cell(row=6+stud_number, column=12).value =
        ↪  points_stud_2_2
269     ws_IA_output.cell(row=6+stud_number, column=13).value =
        ↪  points_stud_3_1
270     ws_IA_output.cell(row=6+stud_number, column=14).value =
        ↪  points_stud_3_2
271     ws_IA_output.cell(row=6+stud_number, column=15).value = points_stud_4
272     ws_IA_output.cell(row=6+stud_number, column=16).value = points_stud_5
273     wb_IA_output.save('IA_Output_stud.xlsx')
```

B Statutory Declaration

I hereby declare that the thesis with title

"Testing Momentum Strategies using Python"

has been composed by myself autonomously and that no means other than those declared were used. In every single case, I have indicated parts that were taken out of published or unpublished work, either verbatim or in a paraphrased manner, as such through a quotation.

This thesis has not been handed in or published before in the same or similar form.

ZÃ¼rich, March 1, 2023 Saskia Senn