**GROUP WORK PROJECT #**2 MScFE 600: FINANCIAL DATA
**GROUP NUMBER:** 435

| FULL LEGAL NAME | LOCATION (COUNTRY) | EMAIL ADDRESS | MARK X FOR ANY NON-CONTRIBUTING MEMBER |
|---|---|---|---|
| Sathaphon Noychin | Thailand | sathaphon.noychin@gmail.com | |
| Ahmad Izzudin | Indonesia | ahmadizzudin41@gmail.com | |
| Hillary Chima Chejieh | Nigeria | contesseration_yahuna@slmail.me | |

| **Statement of integrity:** By typing the names of all group members in the text boxes below, you confirm that the assignment submitted is original work produced by the group (excluding any non-contributing members identified with an "X" above). | |
|---|---|
| **Team member 1** | **Sathaphon Noychin** |
| **Team member 2** | **Ahmad Izzudin** |
| **Team member 3** | **Hillary Chima Chejieh** |

| Use the box below to explain any attempts to reach out to a non-contributing member. Type (N/A) if all members contributed.<br>**Note:** You may be required to provide proof of your outreach to non-contributing members upon request. |
|---|
| (N/A) |

# Step 1 & 2

## 1. LIST

### 1.1. Introduction

Lists are ordered collections of various data types, such as strings, booleans, and other lists. Each element has an index starting from zero. The list data structure is mutable. This means that the list can grow or shrink at will. Modifiability of the list is indicated by adding, removing, or updating items in the list. Square brackets are used to enclose the list. To declare a list (Python Software Foundation, 2022a) :

```
companies = ['Google', 'Facebook', 'Apple', 'Microsoft']
```

This list has 4 elements. We can access each element in the list using its index:

| Input | Output |
|---|---|
| `print(companies[0])`<br>`print(companies[1])` | **`Google`**<br>**`Facebook`** |

**Subsetting a list** can be done by ***slicing***. **Slicing** is used to build a new list from the existing list, followed by syntax: `my_list[start:end]`. The *`start`* index will be included, while the *`end`* index is *not*.

| Input | Output |
|---|---|
| `print(companies[0:2])` | `['Google', 'Facebook']` |

**Manipulating Lists**

| Replace list elements | |
|---|---|
| `companies[1] = 'Amazon'`<br>`print(companies)` | `['Google', 'Amazon', 'Apple', 'Microsoft']` |
| Extend a list | |
| `companies_new = companies + ['Facebook']`<br>`print(companies_new)` | `['Google', 'Amazon', 'Apple', 'Microsoft', 'Facebook']` |
| Delete list elements | |
| `del(companies_new[4])`<br>`print(companies_new)` | `['Google', 'Amazon', 'Apple', 'Microsoft']` |

**List Methods**

| **index():** get the index of the first element of a list that matches its input | |
|---|---|
| `companies = ['Google', 'Facebook', 'Apple', 'Microsoft']`<br>`print(companies.index('Apple'))` | `2` |
| **count():** get the number of times an element appears in a list. | |
| `companies = ['Google', 'Facebook', 'Apple', 'Microsoft']`<br>`print(companies.count('Apple'))` | `1` |
| **append():** add an element to the list it is called on | |
| `companies = ['Google', 'Facebook', 'Apple', 'Microsoft']`<br>`companies.append('Twitter')`<br>`print(companies)` | `['Google', 'Facebook', 'Apple', 'Microsoft', 'Twitter']` |
| **remove():** remove the first element of a list that matches the input | |
| `companies = ['Google', 'Facebook', 'Apple', 'Microsoft']` | `['Google', 'Facebook', 'Apple']` |

| | |
|---|---|
| `companies.remove('Microsoft')`<br>`print(companies)` | |
| **reverse()**: reverse the order of the elements in the list it is called on. | |
| `companies = ['Google', 'Facebook', 'Apple', 'Microsoft']`<br>`companies.reverse()`<br>`print(companies)` | `['Microsoft', 'Apple', 'Facebook', 'Google']` |
| **pop():** remove the last element in the list | |
| `companies = ['Google', 'Facebook', 'Apple', 'Microsoft']`<br>`remove_company = companies.pop(3)`<br>`print(companies)` | `['Google', 'Facebook', 'Apple']` |

## 1.2. Advantages and Disadvantages of a List

**Here are some advantages of using list data structures:**

- Ease of use with many data types.
- Since lists have consecutive elements, it's easy to track unique elements in a list.
- Easy operations such as adding, deleting, and updating items.
- List comprehensions make code easier to read and have many advantages over loops and other methods

**Here are some disadvantages of using list data structures:**

- A limitation of lists is that it is difficult to add items to the beginning of a list, as items can only be added to the end of the list.
- Rotating items in a list is also difficult (geeksforgeeks,2022a).

## 1.3. Study-aid / crib-sheet for best practices with list data structure

**List comprehensions are very useful. This is because you can create the list in one line by specifying a loop .**

```
# Create list comprehension: squares

squares = [i**2 for i in range(0,10)]

squares = [i**2 for i in range(0,10)]

print(squares)

[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

**Stacks and queues** are the two most common uses of list data structures in Python. Stacks and queues are used in many real-world applications and are basically last-in-first-out and first-in-first-out respectively (Python Software Foundation, 2022b).

# 2. DICTIONARY

## 2.1. Introduction

A dictionary in Python is a compound data type that represents a collection of objects. It is a mutable data type that does not allow duplicate data. In addition, a dictionary stores data in a single object with key-value pairs representing each element of the object. Each key (similar to regular keys for locking) is unique, allowing access to insert, read, or replace the corresponding value.

Dictionary keys can be either characters, string literals, or numbers, but values can range from characters, strings, or numbers to arrays or other dictionaries. Basically, we can have dictionaries in dictionaries.

As of Python 3.7, dictionaries are ordered. This means that every element in the dictionary has a specific index. In addition to using keys, we can also use indexes as long as they don't cause ambiguity. Before Python 3.7, dictionaries were unordered.

We can also create a new dictionary by specifying all members as comma-separated key-value pairs in curly braces, like this: exampleDict = { "school": "WQU", "course" : "MScFE" }. This creates a dictionary of two keys (school and course) with their respective values.

Furthermore, we can use the built-in Python function len() to get the number of elements in a dictionary. This function takes a dictionary item as a parameter and returns the number of items as an integer.

To access or get a value in a dictionary, use the key as a reference in square brackets (exampleDict["school"]; which returns "WQU") or use the get() Use functions. As a parameter (exampleDict.get("school"); this returns "WQU")

To update or edit a dictionary value, simply assign the new value to the selected object: exampleDict["course"] = "MScFE 600".

New object (key:value pair) to the dictionary, assign the value to the new key as if we were trying to get the value associated with the key. exampleDict["Batch"] = "July 5th".

To remove an element from a Python dictionary, use the built-in function pop(). ExampleDict.pop(“batch”). After this action, we will no longer be able to access the item using the "Batch" button.

Another convenient way to remove items is the built-in function popItem(). This will remove the last inserted value from the dictionary. We can also use the 'del' keyword to remove elements or delete the dictionary entirely.

del exampleDict["batch"]. This achieves similar results to the pop() function but applying it directly to a dictionary completely removes the dictionary and all objects within it.

Using the clear() function completely clears the dictionary, but not the dictionary itself. That is, all objects are deleted, but the elements still exist in memory.

We can copy a dictionary using the copy() built-in function((Python Software Foundation, 2022c).

## 2.2. Advantages and Disadvantages of a Dictionary

**The advantages of choosing a dictionary as your data structure are:**

- Dictionaries make your code more readable. The code is easy for anyone to understand because the properties are already clearly written out as the keys and associated values need to be written out.
- Creating, modifying, or retrieving values from a dictionary is relatively straightforward.
- Unique key-value relationships simplify modeling and analysis.
- Uses a specific key to access each value instead of searching through a list of items to find a known item, which saves a lot of time.

**The disadvantages of choosing a dictionary as your data structure are:**

- Python dictionaries occupy a lot of memory unlike other data structures. This becomes more of an issue when you have many keys and possibly multiple dictionaries.
- Newer versions of Python have ordered dictionaries, but dictionaries may not be suitable for backward compatibility if the order of the data involved is important (geeksforgeeks,2022b) .

## 2.3. Study-aid / crib-sheet for best practices with dictionaries

Dictionaries are based on hash maps and have unique key-value pairs, so they are best used for unique records.

The usefulness of quick dictionaries is also maximized when dealing with unique, unordered records.

Use built-in functions when performing operations on dictionaries. For some operations, there are some built-in functions that are more efficient than doing them manually. Here are some notable examples:

- update() function. Can be used to merge two dictionaries. For example; suppose you have two dictionaries dict1 and dict2. Then dict1.update(dict2) merges both dictionaries and replaces only similar key values in dict1 with values in dict2.
- The items() function is the fastest way to get and iterate over all key-value pairs.
- The keys() and values() functions, like the items() function, are the fastest way to get and iterate over all keys or values.

# 3. DATAFRAME

## 3.1. Introduction

DataFrame is the data structure under Pandas, a library built on top of Numpy with the goal of facilitating flexible data manipulation. This is a feature that Numpy lacks, as it requires homogenous data types across its data structures, while Numpy maintains an excellent ability to perform arithmetic operations and reductions. A DataFrame is a tabular data structure that contains a collection of columns of various data types. Over time, DataFrames will be integrated with a number of features useful for data science, such as: B. Integration and visualization with common databases and data files (Jake VanderPlas, 2016; Worldquant University, 2022).

| Manipulation Operations | Syntax |
|---|---|
| Creation | - ***From dictionary: dictionary keys are assumed as column names while dictionary values are assumed as rows' content***<br>data = {<br>'state': ['Ohio', 'Ohio', 'Ohio', 'Nevada', 'Nevada'],<br>'year': [2000, 2001, 2002, 2001, 2002],<br>'pop': [1.5, 1.7, 3.6, 2.4, 2.9]<br>}<br>frame = pd.DataFrame(data)<br><br>- ***From csv file (most commonly)***<br>data = pd.read_csv('datafile.csv')<br><br>- ***From other common file types such as excel and sql databases***<br>data = pd.ExcelFile('data.xls')<br><br>import pandas.io.sql as sql<br>data = sql.read_frame('select * from test', con) |
| Subsetting | ***- Select rows 5-10***<br>data.iloc[5:10]<br><br>***- Select rows meeting logical condition, and only specific columns***<br>data.loc[df['pop'] > 2, ['state', 'year']]<br><br>***- Access single value by index***<br>data.iat[2, 3]<br><br>***- Access single value by label***<br>data.at[2, 'state']<br><br>***- Access using query like method***<br>data.query('year < 2002 and pop > 2') |
| Wrangling: Join, Combine | df1 = pd.DataFrame({'x1':["A", "B", "C"], 'x2':[1, 2, 3]}) |

| Manipulation Operations | Syntax |
|---|---|
| | df2 = pd.DataFrame({'x1':["A", "B", "D"], 'x3':[T, F, T]})<br><br>***- Standard Join***<br>*Join matching row from df1 and df2*<br>pd.merge(df1, df2, how='left', on='x1'<br>*All rows in df1 that have a match in df2*<br>df1[df1.x1.isin(df2.x1)]<br><br>***- Reshape***<br>pd.concat([df1, df2]) |

## 3.2. Advantages and Disadvantages of a DataFrame

**Following are some advantages of using DataFrame:**

- High performance DataFrame inherits Numpy's ability to perform matrix arithmetic operations.
- Extensive feature set in subsetting and consolidation. Currently, DataFrames can be created directly from popular data files for big data such as hdf5, pickle, and parquet. It also supports queries from nosql databases such as mongodb. This makes the DataFrame the de facto data structure for data analysis.

**Following are some disadvantages of using DataFrame:**

- Pandas has a steep learning curve. At first, it's easy to grasp the basic concepts. However, the deeper you dig into the library, the steeper the learning curve.
- Panda's ability to handle high-dimensional data is inadequate. It's great for spreadsheet-like data, but has poor support for 3D matrices. In such cases, the user must resort to her Numpy or another library.
- Pandas is popular with Python users, but syntax and support can vary from version to version, making code harder to maintain than more mature languages like R or SAS (geeksforgeeks,2022c).

## 3.3. Study-aid / crib-sheet for best practices with DataFrames

- **Use loc/iloc to replace values:**

Splitting on square brackets is convenient but using iloc/loc is counter-intuitive at first, but using loc/iloc significantly improves code performance. increase. This is because each bracket calls the loc/iloc API once. Using iloc can reduce API calls and execution time.

- **Using vectorization:**

pandas is based on Numpy, which supports vectorization. Therefore, when writing instructions for pandas, using vectorization can help reduce execution time and memory footprint because computations are done in parallel.

- **Use a special pandas library:**

pandas normally uses only one CPU core. Using specialized libraries like Pandaralel to take advantage of multiple CPU cores and facilitate computation can minimize code changes.

# Step 3

See attached Jupyter notebook

## References

Geeksforgeeks,2022a. Advantages and Disadvantages of a List, 8 October.
Geeksforgeeks,2022b. Advantages and Disadvantages of a Dictionary, 8 October.
Geeksforgeeks,2022c. Advantages and Disadvantages of a Dataframe, 8 October.
Jake VanderPlas, 2016. Python Data Science Handbook.
Yves Hilpisch, 2019. Python for Finance 2e : Mastering Data-Driven Finance.
Python Software Foundation, 2022a. List, 9 October.
Python Software Foundation, 2022b. Dictionaryt, 9 October.
Python Software Foundation, 2022c. Dataframe, 9 October.
WorldQuant University (2022) MScFE 600 Financial Data: Group work project 1, 20 September.
WorldQuant University (2022) MScFE 600 Financial Data: Lecture notes, 20 September.