# DD2480 - Refactoring

André Fredriksen
Gustaf Larsson
Noel Tesfalidet
Rafael Bechara
Samuel Greenberg

## 1 Project

**Name:** Pandas

**URL:** link

Python library that provides data structures such as Dataframe and data analysis tools. The library is mostly used for data manipulation, analysis and visualization of data.

## 2 Onboarding experience

For the previous project we used Jabref, a bibliography management tool, but for this lab we wanted to give another project a shot. We were also not set on programming in Java so we searched for projects written in Python. At first we considered PyTorch but in the end we decided on Pandas due to how commonly it is used in Python programming.

Jabref had very detailed documentation on how to fork the repo, what version of JDK was needed and how to get started on the project using Intellij. It also included step by step instructions on how to compile, run the project and run the tests. It even included detailed instructions on how to set up a a style checker that they recommended. This meant that after following the step by step instructions, the project worked without any issues.

Pandas had decent documentation on how to get started with the project but it was nowhere close to the detail that Jabref's documentation had. The setup guide included general steps like "install a C compiler and "install an isolated environment" but it did not include a detailed step by step guide. Most of the general steps were pretty easy to do, for example installing a C compiler or running a command to compile the project. The guide recommended using mamba and then linked to mamba's installation guide. Unfortunately that installation guide was very unclear and setting up an isolated environment was therefore a major struggle for everyone in the group.

## 3 Effort spent

For each team member, how much time was spent in

For setting up tools and libraries (step 4), enumerate all dependencies you took care of and where you spent your time, if that time exceeds 30 minutes.

| Category | André | Gustaf | Noel | Rafael | Samuel |
|---|---|---|---|---|---|
| Plenary discussions/meetings | 3 | 3 | 3 | 4 | 4 |
| Reading documentation | 1 | 1 | 2 | 1 | 1 |
| Configuration and setup | 2 | 0,5 | 3 | 3 | 2 |
| Analyzing code / output | 8 | 10 | 8 | 6 | 8 |
| Writing documentation / report | 3 | 4 | 4 | 1 | 2 |
| Writing code | 3 | 5 | 5 | 3 | 2 |
| Running code | 1 | 1 | 1 | 1 | 1 |

## 4 Overview of issue(s) and work done

### 4.1 Issue #57594

#### 4.1.1 Overview of issue

**Title:** BUG: df.plot makes a shift to the right if frequency multiplies to $n > 1$

**Summary:** Dataframe.plot with times generated using *period_range*() works well for frequency with $n = 1$ but for frequencies where $n > 1$ all times are shifted to the right (where n is a time unit, f.eg. minutes, hours, days, years, quarters etc.).

For example, when stating that the period should start at 00:00 with 4 measurements, at a frequency of 3 hours and the values [0,1,0,1] this is the actual and expected plots:

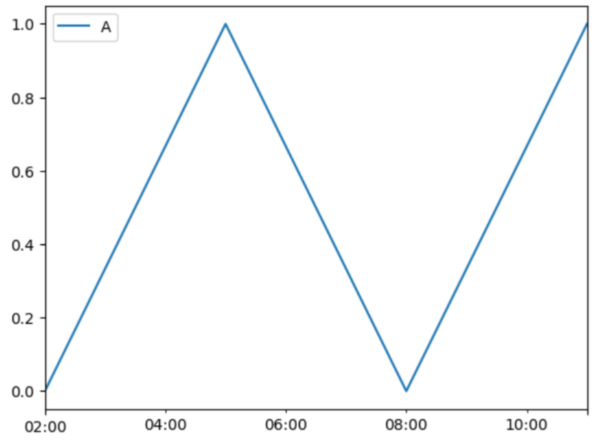

Figure 1: Actual (bugged) plot.

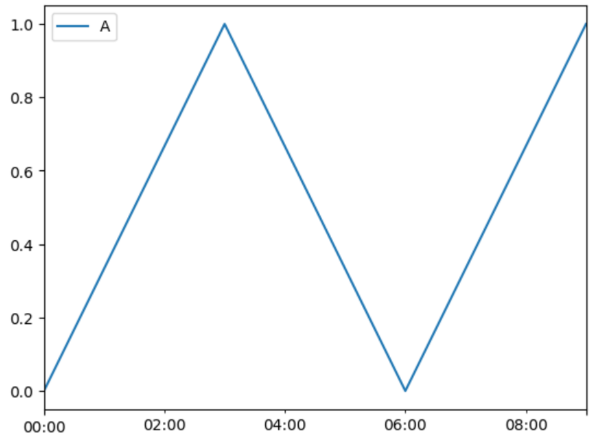

Figure 2: Expected plot.

#### 4.1.2 Work Done

The cause of this bug is that by default all elements are generated to align to the end of a period instead of at the start and the period technically does not end at the start of the last period but rather just before the period subsequent to the last period. To fix this problem we therefore passed a parameter stating that the elements should be aligned to the start of the period instead of the end.

## 4.2 Issue #48188

### 4.2.1 Overview of issue

**Title:** #48188 BUG: Using NamedTuples with .iloc fails

**Summary:** Dataframe.iloc can be used for indexing position in the Dataframe using integers only. Some examples are:

- An integer, e.g. 5.
- A list or array of integers, e.g. [4, 3, 0].
- A tuple of integers, e.g. ([4, 3).
- A slice object with ints, e.g. 1:7.
- A boolean array.

However, with namedtuples it doesn't work due to it not being classified as an integer due to the "named" attribute that gets included.

```python
import pandas as pd
import numpy as np

# A df with two level MultiIndex
df = pd.DataFrame(np.arange(120).reshape(6, -1))

# Indexing with a normal tuple works as expected:
# normal subset
print(df.iloc[1, 2])  # <- Works
print(df.iloc[(1, 2)])  # <- Works

# Now the same with a named tuple
from collections import namedtuple

indexer_tuple = namedtuple("Indexer", ["x", "y"])

# simple subset with named tuple
print(df.iloc[indexer_tuple(x=1, y=2)])  # <- Does not work
# Raises:
# TypeError: Cannot index by location index with a non-integer key
# Converting to a tuple works again
print(df.iloc[tuple(indexer_tuple(x=1, y=2))])  # <- Works

```

Listing 1: Example that produces the bug.

The above code snippet shows a reproducible example where all four uses of iloc() should behave identical but the third usage in this case would raise an error stating that the argument is not an integer.

#### 4.2.2 Work Done

The iloc() function would call a private method __getitem__(self, key) with would check if the given key is a tuple and otherwise, it would call _getitem_axis(self, key, axis: AxisInt) which would go through several conditions and check for other possible indexing types. The solution was to add another condition that checks if the key is a namedtuple, in that case, cast it to a tuple and call the __getitem__(self, key) again with the casted key.

## 5 Requirements for the new feature or requirements affected by functionality being refactored

### 5.1 Issue #57594

1. The first date in the list given by $period\_range(start, freq, periods)$ should be the same as $start$.
2. The time between two consecutive elements in the list given by $period\_range(start, freq, periods)$ should be equal to freq.
3. The number of elements in the list given by $period\_range(start, freq, periods)$ should be equal to $periods$.
4. Each point in the plot should correspond to a date in the DataFrame meaning there should only be points on the dates in the DataFrame and there should be one point on each of the dates in the DataFrame.

The modified test in tests/indexes/period/test_period_range.py tests requirement 1,2 & 3 while the two tests written in tests/plotting/test_datetimelike.py tests requirement 4.

### 5.2 Issue #48188

1. Using a namedtuple as index for a DataFrame should not throw an error
2. Using a namedtuple as index for a DataFrame produce the same return value as a normal tuple if the values of the tuples are the same

The test added to tests/indexing/test_iloc.py tests both requirements.

## 6 Code changes

### 6.1 Patches

Pull request for issue #57594
Pull request for issue #48188

# 7 Test results

## Running tests on cloned project

Pandas has  230000 tests, but not all of them were relevant for the issues we chose.

There were 213 tests related to plotting data where one of the axis should be dates, all of them in *tests/plotting/test_dateTimeLike.py* which were related to plotting data where at least one of the inputs is dates. 209 of the date plotting tests passed. The 4 tests that failed were marked as expecting to fail, 2 due to API changes in 3.6.0 and the other 2 were linked to open issues unrelated to the issue we were working on.

There were 467 tests that tested the Period class, all of them in the directory *tests/indexes/period*. All tests passed.

For testing indexing there were 4464 test where 215 of them were specifically for the iloc() function. These tests can be found in *tests/indexing/test_iloc.py*. All of these tests passed or failed as an expected failure. This indicates that the bug described in the issue likely isn't being tested in the current test suite.

Date Plotting logs
Period logs
iloc logs

## Adding tests

In tests/plotting/test_datetimelike.py we added 2 new tests which checked whether the input dates were equal to the dates in the plot when the input dates were generated with *period_range*() and *date_range*() respectively. These two tests were then run with 12 different frequencies: 20 seconds, 3 seconds, 60 minutes, 5 minutes, 7 hours,4 days, 8 weeks, 11 months, 2 quarters, 1 year, 3 years & 10 years. When running the tests the test using *date_range*() passes all with all frequencies while *period_range*() only passes with the frequency of a year.

In tests/indexes/period/test_period_range.py no completely new tests were added but one existing test was modified to run with new frequencies. Previously the test was only executed with the frequencies: 1 week, 1 day, 1 quarter & 1 year. Therefore, we added the frequencies: 1 hour, 7 hours, 15 days, 1 month, 3 months, 8 years, 20 seconds & 2 quarters. This test passed with all parameters, indicating that the bug was related to plotting and not the test_period_range function.

In tests/indexing/test_iloc.py a new test was added that reproduced the bug from the issue and rerunning all tests for iloc() showed one additional test which failed (which is to be expected). The test works by calling the iloc() function with various integer key types such as a tuple and array. It then also calls the iloc() with a namedtuple and thereafter compares the output of each iloc call to ensure that they are the same (since the same integer value of key was used). The test could fail either if the call with a namedtuple throws an error (which is the current bug) or if any of the calls to iloc() don't return the same value as the other calls.

iloc logs with new test added

### After the bugfix

After the parameter was changed no new tests failed and the test using *period_range*() now passed for all frequencies.

With the new condition added for the iloc() bug the test that previously failed, now succeeded. No tests that previously passed now failed.

Date Plotting logs
Period logs
iloc logs

## 8 UML class diagram and its description

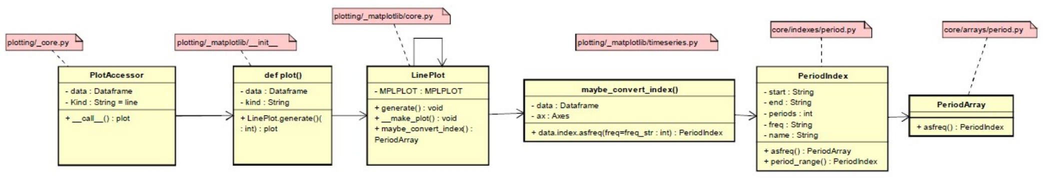

Figure 3: UML Diagram showing the plotting execution for period_range

### 8.1 Key changes/classes affected

Regarding issue #48188, the added code follows the same syntax code style as the rest of the project. Similarly for the added test it also follows the same structure of how their test suite is built. There isn't much to be said about design patterns for the issue as the added code itself doesn't design anything *new* it simply expands on an existing design.

## 9 SEMAT kernel

### 9.1 Issue #48188

**Benefits:**

- Oppotunity: Issue #48188 addresses the specific need of fixing the bug which makes it so that namedtuples aren't supported for the iloc() function.
- Requirements: The commit aligns with the project requirements as the commit fixes a reported issue.
- Software System: All previous tests within the same submodules of pandas still pass when tested, the newly added test also passes.

**Drawbacks:**

- Software System: This solution might limit the extensibility of the project as this issue adds a type-checking condition specifically for namedtuple which might incentivize similar solutions to similar issues in the future which would increase the complexity of the code.

**Limitations:**

- Oppotunity: By focusing on the specific problem with namedtuple, there might be a missed opportunity to address the underlying issue in a more generalized manner that could accommodate not only namedtuple but also other types that might benefit from similar handling.
- Software System: Adding type-specific logic can introduce maintenance challenges and potential performance implications. Over time, the accumulation of such fixes can lead to a codebase that is harder to understand, test, and optimize.

## 9.2 Issue #57594

**Benefits:**
The SEMAT kernel states that it is good practice to shape the system so that it is easy to develop, change, and maintain, and so that it can cope with current and expected future demands. The unit tests implemented to cover issue #57594 achieve this in two ways. They adhere to code formatting standards of Pandas which means that they are easy for other developers to understand and edit. Secondly, part of the test logic that checks that the x-axis of a *DataFrame.plot* is correct, was placed in a new general function in a file called *common.py* containing other helper functions. This way the function can be used for other tests making it easier to develop, change, and maintain.

**Drawbacks:**
A potential drawback of the solution to the bug is that it solves the issue by passing a parameter in a function call instead of changing the default value of that parameter in the function declaration. This means that other paths of the code may reach the function and cause a bug. The reason we chose this approach is that there is also a risk that changing the default value in the function declaration could have unintended consequences. There may be cases in which the default is correctly set. Pandas is very large and we did not have time to investigate this fully.

**Limitations:**
Our changes solve the issue but there is a possibility that there are similar issues related to other data types of Pandas. More specifically the bug was identified for *LinePlot* but other plot classes could also contain the same bug causing incorrect plotting.

# 10 Ecosystem

## 10.1 Where do you put the project that you have chosen in an ecosystem of open-source and closed-source software? Is your project (as it is now) something that has replaced or can replace similar proprietary software? Why (not)?

Pandas started as an open-source project and does not directly replace any proprietary software. There are other tools that has similar functionality, for example, Excel and Matlab. A huge advantage of using Pandas instead of Excel or Matlab though is that it is much easier to integrate data manipulation and visualization into already existing software since Pandas is a Python package while Matlab and Excel are specialized software made specifically for data manipulation and visualization.

One of the key advantages of Pandas is its open-source nature, which not only makes it accessible to a wider audience but also encourages a collaborative community-driven effort to improve it. This means that users can improve pandas on their own, and add their own ideas whereas application like Excel, Matlab only has the features that people from their company have come up with. This also results in Pandas constantly improving and evolving, not needing to wait for a limited team to deploy new releases on top of the said team having a limited perspective on improvements.

Furthermore, Pandas is built using Python which is used in lots of different areas like data science, machine learning, and web development. Because of this, Pandas works well with other Python tools and programs. This makes it easier for people to use Python to do all kinds of data analysis tasks. Pandas is also doing well in academia since it is open source and students can access this tool without paying for licensing.

Closed-source software, on the other hand, often doesn't play as nicely with other programs because they use their own special file formats and ways of doing things. This can make it harder to use them together with other tools and systems. So, Pandas has an advantage because it's part of the big Python family and can work smoothly with lots of other programs.

While Excel and Matlab may have their strengths in certain use cases, they often fall short when it comes to handling large-scale data manipulation and analysis tasks. Pandas, on the other hand, is specifically designed to efficiently handle large datasets, thanks to its underlying implementation in the Python programming language and optimized data structures. Lastly, Excel is still widely used today which is most likely due to alternatives like Matlab or Pandas requiring programming experience. Whereas Excel is easier for nontechnical people to use. However, the reason many like pandas is because it is more technical which allows for more flexibility. Pandas is most efficient in scenarios that need more complex analysis, and integration with many other machine learning tools for example, which are often difficult to get in the closed source format.

# 11 Overall experience

## 11.1 What are your main take-aways from this project? What did you learn?

The main take-away of this assignment is how long it takes to understand and make changes to complex code that others have written. When the reason for the bug was localized the only thing that had to be changed was changing a parameter to a function call but finding that error took a lot of time. Localizing and understanding the relevant code before making changes is crucial to fix a bug without creating a new one but that takes a lot of time in a project as large and complex as Pandas.

Another take-away is that just because an issue is open does not necessarily mean that it needs fixing. Pandas have over 3000 issues open and there were multiple times that we chose an issue to work on only to then realise that the code mentioned in the issue was already refactored or fixed. If we were to redo this project we would definitely contact one of the moderators to help find suitable issues.

## 11.2 How did you grow as a team, using the Essence standard to evaluate yourself?

We are still in the stage ”Formed” as the team fulfills the checklist for the stage. However, it was more difficult to efficiently distribute the work to each team member for this assignment when compared to the previous one. The reason for this was the nature of the project and not how we worked as a team. The first step involved inspecting existing test coverage and expanding it to reveal the bug. This part could be serialized quite well but when it came to finding the bug it was more difficult to divide the work, leading to multiple members working together on the same task. Otherwise, we communicate well and have begun to know each other and work fairly efficiently as a team. It would, however, take more time, and probably more structured leadership, for the group to become a more efficient working unit and move on to the next stage ”Collaborating”.