Report for Assignment 4

Marcus Jakobsson
Toto Roomi
Rémi Grasset
Kerem Robin Yurt
Anton Bölenius

March 2024

1 Project

Name: Pandas
URL: https://github.com/pandas-dev/pandas
Pandas is a large, widely used graphing and calculation tool in python. Often used in Jupyter notebooks.

2 Onboarding Experience

We chose to move on from Jabref to Pandas. The documentation for Pandas was very detailed and instructive. Some tests were failing, however the main branch was also failing tests on the 4 latests commits. Jabrefs documentation was slightly easier to navigate than Pandas though, this is due to the sheer scale of Pandas. Setting up a python virtual environment with an editable version of the cloned pandas was tricky in the beginning.

3 Effort Spent

For each team member, how much time was spent in:

Time distribution	Robin	Anton	Toto	Remi	Marcus
discussions/meetings	3	3	3	3	3
choosing project/issue	4	4	4	4	4
reading documentation	2	2	2	2	2
configuration and setup	6	5	5	4	5
writing documentation	4	1	6	3	5
writing code	1	3	0	1	2
running code	2	2	0	3	2
Total (h)	**22**	**20**	**20**	**20**	**23**

4 Overview of Issue(s) and Work Done

Title: BUG: to_json does not preserve nanosecond precision of DatetimeIndex with time zone information
URL: bug report , our fix

It was difficult finding a bug as the vast majority we found of reported bugs were either not bugs at all, solved or pending review from repo admins. In the end we found the linked bug, it was a problem where nanosecond precision was lost somewhere in the execution. The majority of time on the issue was spent debugging and locating the bug. The problem was that the piece of data was forwarded through both python and C files, meaning any number of small typos could have caused the bug. Finally, After exhaustive investigation, we traced the problem down to a the file "pandas/_libs/src/datetime/pd_datetime.c" where the function convert_pydatetime_to_datetimestruct() did not handle nanoseconds even though the rest of the code had made space for it. In our solution we added a handler for nanoseconds.

5 Requirements for the New Feature or Requirements Affected by Functionality Being Refactored

Requirements:

- Nanosecond information is retained when creating and reading dataframes that have been saved as Json objects.
- Ensure the fix works with tests

6 Code Changes

6.1 Patch

`git diff ddc3144 682ba46`
We added the following lines to the afformationed file on line 74:
"`if(PyObjec_HasAttrString(obj, "nanosecond")){`
`out − > ps = 1000 * PyLong_AsLong(PyObject_GetAttrString(obj, "nanosecond"));`
}"
We also created a test that makes sure nanoseconds are conserved, even when a timezone is specified.

7 Test Results

Before writing our fix, all the already written test passbut our new test does not.

```
==================================== 1 failed, 445 passed, 28 xfailed in 6.17s ====================================
```

```
=================================== FAILURES ===================================
_____________________ TestPandasContainer.test_tz_nano_datetimes _____________________
>           raise_assert_detail(obj, msg, left, right, index_values=index_values)
E           AssertionError: DataFrame.iloc[:, 0] (column name="date") are different
E
E           DataFrame.iloc[:, 0] (column name="date") values are different (100.0 %)
E           [index]: [0]
E           [left]:  [1704067200000000000]
E           [right]: [1704067200000000001]

pandas\_testing\asserters.py:684: AssertionError
----- generated xml file: C:\Users\RG\Documents\KTH\SE\lab4\pandas\test-data.xml ------
================================ slowest 30 durations ================================

(3 durations < 0.005s hidden.  Use -vv to show these durations.)
============================== short test summary info ==============================
FAILED pandas/tests/io/json/test_pandas.py::TestPandasContainer::test_tz_nano_datetimes
- AssertionError: DataFrame.iloc[:, 0] (column name="date") are different
================================== 1 failed in 0.28s ==================================
(pandas-dev) PS C:\Users\RG\Documents\KTH\SE\lab4\pandas>
```

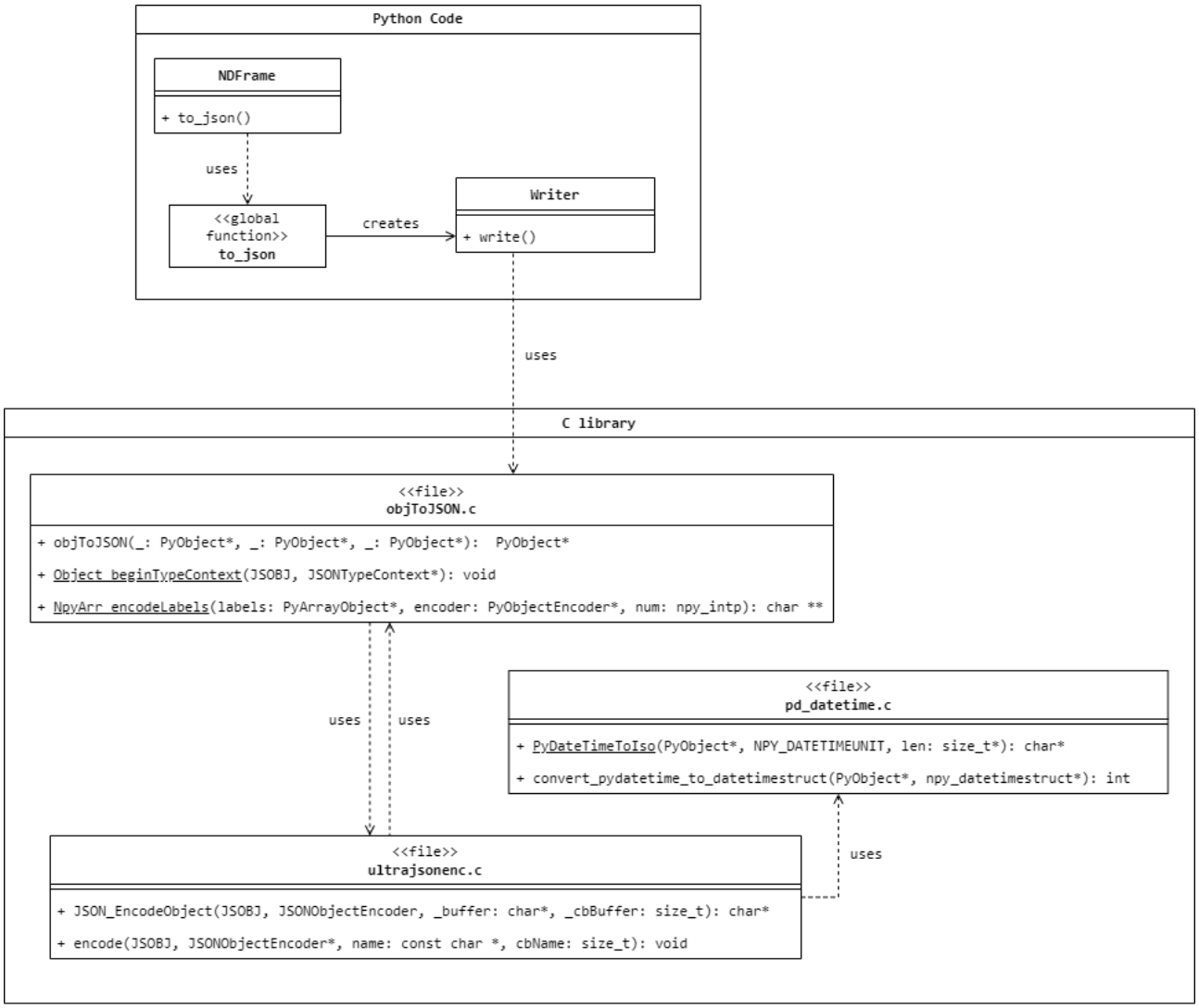
After adding our fix, all the tests pass including our our new test that converts a Dataframe with a Series object containing datetime with timezone and nanosecond precision.

```
==================================== 446 passed, 28 xfailed in 6.10s ====================================
```

```
================================ test session starts ================================
platform win32 -- Python 3.10.13, pytest-8.0.1, pluggy-1.4.0
PyQt5 5.15.9 -- Qt runtime 5.15.8 -- Qt compiled 5.15.8
rootdir: C:\Users\RG\Documents\KTH\SE\lab4\pandas
configfile: pyproject.toml
plugins: anyio-4.3.0, hypothesis-6.98.10, cov-4.1.0, cython-0.2.1, qt-4.4.0, xdist-3.5.0
collected 1 item

pandas\tests\io\json\test_pandas.py .

----- generated xml file: C:\Users\RG\Documents\KTH\SE\lab4\pandas\test-data.xml ------
================================ slowest 30 durations ================================

(3 durations < 0.005s hidden.  Use -vv to show these durations.)
================================== 1 passed in 0.09s ==================================
```

8 UML Class Diagram and Its Description

The feature we worked on is using a Python interface to call a function from an internal C library. Since C is not an object-oriented programming language, it is not possible to create a rigorous UML class diagram. Thus, the following diagram is only inspired by UML class diagram. In the parts regarding C code, files are modeled as classes would be.



9 Self-assesment

Current state according to the Essence standard:
The team has a good dynamic and the current state would be in between the two states collaborating and performing. For the team to be at performing completely one more assignment would lead us to that. The team has a good communication where we are honest and open. In the team we have built a trust for each other where we believe in our teammates. In the team we are setting up requirements that we have deadlines for and so far everyone has performed before the deadline. In the team we are quick and helpful towards each other, especially when it comes to a bigger change that has to be made e.g. change of project.

Was the self-assessment unanimous? Any doubts about certain items?
The self-assessment was unanimous, we have developed a good team structure throughout these four assignments. There is only one small issue within the team and that is that sometimes we tend to wait a bit for the start of a assignment which sometimes can generate stress.

How have you improved so far?
The team has shown great improvement with tools such as github and the features that exists there. We are also better at communicating when assistance is needed. This way the often time consuming process of learning new tools only has to be done by one member, who then can share the knowledge with the others.

Where is potential for improvement?
There is one thing that the team would need to improve which would come by itself with more assignments. When the team has divided tasks the group works with high efficiency, but the start for an assignment can sometimes take some time since we have to pick tasks to finish. This small issue can be due to the fact that it is the first time the group members contributes to an open-source project.

10 Overall Experience

What are your main take-aways from this project?
Our main take-away is that contributing to a large project doesn't have to be a scary and overwhelming task but rather, much like university assignments, just a small problem you have to solve using a general accepted convention. The community is open to discuss and give feedback on the pull requests, which is a great opportunity to contribute and also evolve.

What did you learn?
We've learned how to create tests, document and execute them in a large python project using "pytest". Additionally we've learned the convention used in solv-

ing issues on github. For instance by writing "take" on issues you'd like to get assigned to.

11 P+

Criteria 1 - Architecture

Pandas is a library in Python for data analysis and manipulation, utilized for its efficiency and intuitiveness in handling tabular data. Developed initially in 2008 at AQR Capital Management and open-sourced in 2009, pandas have grown with contributions from a global community, underscored by its sponsorship from NumFOCUS since 2015.[1] This evolution highlights its significance within the data science field, aiming to enhance the open-source ecosystem for data analysis. The architecture of pandas is designed around two main data structures: Series and DataFrame.[1][2]

- Series: is a one-dimensional array-like structure designed to store any data type, including integers, strings, floating-point numbers, and more, each accompanied by a data label or index.
- DataFrame: represents a two-dimensional, table-like structure that can hold data of various types across columns, similar to a SQL table or an Excel spreadsheet. DataFrames support both row and column indices, allowing for sophisticated data manipulation and indexing capabilities.

These data structures are built on top of the NumPy library, benefiting from its high-performance array-computing features. Furthermore, critical paths within pandas utilize Cython or C for performance optimization, ensuring efficient processing of large datasets.

Pandas serve a dual purpose, primarily to facilitate easy and intuitive manipulation of structured data and to provide a robust foundation for sophisticated data analysis.[1][2] Its design goals include:

- Data Manipulation: Offering comprehensive tools for cleaning, transforming, filtering, and aggregating data, making it easier to prepare data for analysis or visualization.
- Data Analysis: Enabling complex operations like group-by, merge, join, and time-series analysis with minimal coding, thereby accelerating the analytical workflow.
- Input/Output: Providing a wide range of functionalities to read from and write to various data formats, including CSV, Excel, SQL, JSON, and HDF5, allowing data scientists to easily interchange data between different sources and formats.

Through its powerful data structures and comprehensive toolset, pandas aim to be the fundamental high-level building block for doing practical, real-world data analysis in Python. It aspires to be accessible, flexible, and efficient, making it an indispensable tool in the collection of data scientists and analysts across various domains, including finance, neuroscience, economics, and more.

Criteria 4 - Clean Patch

The patch can be found using: `git diff ddc3144 682ba46`

```
diff --git a/pandas/_libs/src/datetime/pd_datetime.c b/pandas/_libs/src/datetime/pd_datetime.c
index 19de51be6e..1a45c6883f 100644
--- a/pandas/_libs/src/datetime/pd_datetime.c
+++ b/pandas/_libs/src/datetime/pd_datetime.c
@@ -71,7 +71,9 @@ static int convert_pydatetime_to_datetimestruct(PyObject *dtobj,
   out->min = PyLong_AsLong(PyObject_GetAttrString(obj, "minute"));
   out->sec = PyLong_AsLong(PyObject_GetAttrString(obj, "second"));
   out->us = PyLong_AsLong(PyObject_GetAttrString(obj, "microsecond"));
-
+  if (PyObject_HasAttrString(obj, "nanosecond")) {
+    out->ps = 1000 * PyLong_AsLong(PyObject_GetAttrString(obj, "nanosecond"));
+  }
   if (PyObject_HasAttrString(obj, "tzinfo")) {
     PyObject *offset = extract_utc_offset(obj);
     /* Apply the time zone offset if datetime obj is tz-aware */
diff --git a/pandas/tests/io/json/test_pandas.py b/pandas/tests/io/json/test_pandas.py
index db120588b2..73e52b35dc 100644
--- a/pandas/tests/io/json/test_pandas.py
+++ b/pandas/tests/io/json/test_pandas.py
@@ -1216,6 +1216,26 @@ class TestPandasContainer:
         s_naive = Series(tz_naive)
         assert stz.to_json() == s_naive.to_json()

+    def test_tz_nano_datetimes(self):
+        df = DataFrame(
+            {
+                "date": Series(
+                    [
+                        datetime.datetime(
+                            2024, 1, 1, 0, 0, 0, 000000, tzinfo=datetime.timezone.utc
+                        )
+                    ]
+                )
+            }
+        )
+        df.date = df.date + np.timedelta64(1, "ns")
+        buf = StringIO()
+        df.to_json(buf, date_unit="ns", orient="columns", date_format="iso")
+        buf.seek(0)
+        tm.assert_frame_equal(
+            read_json(buf), df, check_index_type=False, check_dtype=False
+        )
+
     def test_sparse(self):
         # GH4377 df.to_json segfaults with non-ndarray blocks
         df = DataFrame(np.random.default_rng(2).standard_normal((10, 4)))
```

- **a)** We have no obsolete code.
- **b)** We didn't add any debug code.
- **c)** We didn't add any unnecessary white spaces.

Criteria 5 - Patch Accepted

For now we are waiting on the PR to get accepted. PR Link

Criteria 6 - Work Carried out

Customer

- **Opportunity**
  The opportunity alpha covers the reason why the software is developed and also represents the stakeholders needs. The software is developed with the aim to assist data analysis and manipulation.

- **Stakeholders**
  Pandas is used to visualize data in many fields of research, from data science and biology to humanities like psychology. It's commonly used together with Jupyter notebooks. Therefor the stakeholders consist of everything from researchers, R&D groups in companies, students and so on. The core group of developers of pandas are also stakeholders.

Solution

- **Requirements**
  Pandas lists the highlights of the library in their website [1]. These can be translated to the requirements of the system. DataFrames are stated to be fast and efficient objects for data manipulation. While fast and efficient, it must also be reliable. This is what our bug fix solves.

- **Software System**
  Pandas is an open source, BSD-licensed, Python library.

Benefits of our work

We worked on an issue where there was a problem related to a date time precision. The reported issue was that a person wanted to save a data frame as json which later was loaded as a data frame again. Ideally it should result in the same thing as the original, but there existed a problem where the json string had microseconds as precision and not nanoseconds which was the asked precision. Our work carried out the problem that was related to the precision. With our work there are a lot of stakeholders that could be satisfied with a correct data frame when using nanoseconds as precision. Our work also covers the alpha *"requirements"* since the issue is now corrected and the system will work as intended for the stakeholders.

Drawbacks of our work

While our fix solves a problem and tests for it, it may break future additions to the library that we cannot account for. Additionally, at the time of our project, several tests were not passing in the main branch. Since DataFrames are an integral part of Pandas, it is unclear whether our fix keeps these tests broken. There could also be old usages of the code that were taking into account the

bug and adding the required precision manually. This old code would then be dysfunctional after our fix.

Limitations of our work

Our fix only spans about 20 hours of work in a large project that has been in development since 2008 [1]. While our fix is small, DataFrames permeate throughout the library and affect many parts of Pandas. It would be unfeasible for us to onboard the entire project and learn about every library. Ideally we would either learn about everything DataFrames affect or speak to a senior developer about potential pit falls. Thus our project is limited by the scope of the assignment in the course.

Criteria 7 - Extraordinary

In assignment 3 we had the opportunity to work on a open-source project where we implemented a coverage tool. This time we got the opportunity to contribute in the source code and help the community resolve some issues. Contributing to an open-source project is new for most of us and having an assignment that introduces to that is a good first step. We chose to work on pandas which is a Python library. The library is towards data analysis and closely related areas. Most of the group members has at least once used the library and was glad to contribute to a project they have used before. With all that said we are proud of our contribution to pandas and this assignment has given us a push towards contributing to open-source projects.

References

[1] *Pandas - Python Data Analysis Library*. [Online]. Available: https://pandas.pydata.org/about/ (visited on 03/08/2024).

[2] *Pandas (software)*, en, Page Version ID: 1200717557, Jan. 2024. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Pandas_(software)&oldid=1200717557 (visited on 03/08/2024).