- Dylan Dmitri Gray +
- Eric Boxer +
- Eric Chea
- Erik +
- Erik Nilsson +
- Fabian Haase +
- Fabian Retkowski
- Fabien Aulaire +
- Fakabbir Amin +
- Fei Phoon +
- Fernando Margueirat +
- Florian Müller +
- Fábio Rosado +
- Gabe Fernando
- Gabriel Reid +
- Giftlin Rajaiah
- Gioia Ballin +
- Gjelt
- Gosuke Shibahara +
- Graham Inggs
- Guillaume Gay
- Guillaume Lemaitre +
- Hannah Ferchland
- Haochen Wu
- Hubert +
- HubertKl +
- HyunTruth +
- Iain Barr
- Ignacio Vergara Kausel +
- Irv Lustig +
- IsvenC +
- Jacopo Rota
- Jakob Jarmar +
- James Bourbeau +
- James Myatt +
- James Winegar +

- Jan Rudolph
- Jared Groves +
- Jason Kiley +
- Javad Noorbakhsh +
- Jay Offerdahl +
- Jeff Reback
- Jeongmin Yu +
- Jeremy Schendel
- Jerod Estapa +
- Jesper Dramsch +
- Jim Jeon +
- Joe Jevnik
- Joel Nothman
- Joel Ostblom +
- Jordi Contestí
- Jorge López Fueyo +
- Joris Van den Bossche
- Jose Quinones +
- Jose Rivera-Rubio +
- Josh
- Jun +
- Justin Zheng +
- Kaiqi Dong +
- Kalyan Gokhale
- Kang Yoosam +
- Karl Dunkle Werner +
- Karmanya Aggarwal +
- Kevin Markham +
- Kevin Sheppard
- Kimi Li +
- Koustav Samaddar +
- Krishna +
- Kristian Holsheimer +
- Ksenia Gueletina +
- Kyle Prestel +
- LJ +

- LeakedMemory +
- Li Jin +
- Licht Takeuchi
- Luca Donini +
- Luciano Viola +
- Mak Sze Chun +
- Marc Garcia
- Marius Potgieter +
- Mark Sikora +
- Markus Meier +
- Marlene Silva Marchena +
- Martin Babka +
- MatanCohe +
- Mateusz Woś +
- Mathew Topper +
- Matt Boggess +
- Matt Cooper +
- Matt Williams +
- Matthew Gilbert
- Matthew Roeschke
- Max Kanter
- Michael Odintsov
- Michael Silverstein +
- Michael-J-Ward +
- Mickaël Schoentgen +
- Miguel Sánchez de León Peque +
- Ming Li
- Mitar
- Mitch Negus
- Monson Shao +
- Moonsoo Kim +
- Mortada Mehyar
- Myles Braithwaite
- Nehil Jain +
- Nicholas Musolino +
- Nicolas Dickreuter +

- Nikhil Kumar Mengani +
- Nikoleta Glynatsi +
- Ondrej Kokes
- Pablo Ambrosio +
- Pamela Wu +
- Parfait G +
- Patrick Park +
- Paul
- Paul Ganssle
- Paul Reidy
- Paul van Mulbregt +
- Phillip Cloud
- Pietro Battiston
- Piyush Aggarwal +
- Prabakaran Kumaresshan +
- Pulkit Maloo
- Pyry Kovanen
- Rajib Mitra +
- Redonnet Louis +
- Rhys Parry +
- Rick +
- Robin
- Roei.r +
- RomainSa +
- Roman Imankulov +
- Roman Yurchak +
- Ruijing Li +
- Ryan +
- Ryan Nazareth +
- Rüdiger Busche +
- SEUNG HOON, SHIN +
- Sandrine Pataut +
- Sangwoong Yoon
- Santosh Kumar +
- Saurav Chakravorty +
- Scott McAllister +

- Sean Chan +
- Shadi Akiki +
- Shengpu Tang +
- Shirish Kadam +
- Simon Hawkins +
- Simon Riddell +
- Simone Basso
- Sinhrks
- Soyoun(Rose) Kim +
- Srinivas Reddy Thatiparthy ( ) +
- Stefaan Lippens +
- Stefano Cianciulli
- Stefano Miccoli +
- Stephen Childs
- Stephen Pascoe
- Steve Baker +
- Steve Cook +
- Steve Dower +
- Stéphan Taljaard +
- Sumin Byeon +
- Sören +
- Tamas Nagy +
- Tanya Jain +
- Tarbo Fukazawa
- Thein Oo +
- Thiago Cordeiro da Fonseca +
- Thierry Moisan
- Thiviyan Thanapalasingam +
- Thomas Lentali +
- Tim D. Smith +
- Tim Swast
- Tom Augspurger
- Tomasz Kluczkowski +
- Tony Tao +
- Triple0 +
- Troels Nielsen +

- Tuhin Mahmud +
- Tyler Reddy +
- Uddeshya Singh
- Uwe L. Korn +
- Vadym Barda +
- Varad Gunjal +
- Victor Maryama +
- Victor Villas
- Vincent La
- Vitória Helena +
- Vu Le
- Vyom Jain +
- Weiwen Gu +
- Wenhuan
- Wes Turner
- Wil Tan +
- William Ayd
- Yeojin Kim +
- Yitzhak Andrade +
- Yuecheng Wu +
- Yuliya Dovzhenko +
- Yury Bayda +
- Zac Hatfield-Dodds +
- aberres +
- aeltanawy +
- ailchau +
- alimcmaster1
- alphaCTzo7G +
- amphy +
- araraonline +
- azure-pipelines[bot] +
- benarthur91 +
- bk521234 +
- cgangwar11 +
- chris-b1
- cxl923cc +

- dahlbaek +
- dannyhyunkim +
- darke-spirits +
- david-liu-brattle-1
- davidmvalente +
- deflatSOCO
- doosik_bae +
- dylanchase +
- eduardo naufel schettino +
- euri10 +
- evangelineliu +
- fengyqf +
- fjdiod
- fl4p +
- fleimgruber +
- gfyoung
- h-vetinari
- harisbal +
- henriqueribeiro +
- himanshu awasthi
- hongshaoyang +
- igorfassen +
- jalazbe +
- jbrockmendel
- jh-wu +
- justinchan23 +
- louispotok
- marcosrullan +
- miker985
- nicolab100 +
- nprad
- nsuresh +
- ottiP
- pajachiet +
- raguiar2 +
- ratijas +

- realead +
- robbuckley +
- saurav2608 +
- sideeye +
- ssikdar1
- svenharris +
- syutbai +
- testvinder +
- thatneat
- tmnhat2001
- tomascassidy +
- tomneep
- topper-123
- vkk800 +
- winlu +
- ym-pett +
- yrhooke +
- ywpark1 +
- zertrin
- zhezherun +

5.4 Version 0.23

5.4.1 What's new in 0.23.4 (August 3, 2018)

This is a minor bug-fix release in the 0.23.x series and includes some small regression fixes and bug fixes. We recommend that all users upgrade to this version.

Warning: Starting January 1, 2019, pandas feature releases will support Python 3 only. See Dropping Python 2.7 for more.

What's new in v0.23.4

- *Fixed regressions*
- *Bug fixes*
- *Contributors*

Fixed regressions

- Python 3.7 with Windows gave all missing values for rolling variance calculations (GH21813)

Bug fixes

Groupby/resample/rolling

- Bug where calling `DataFrameGroupBy.agg()` with a list of functions including `ohlc` as the non-initial element would raise a `ValueError` (GH21716)
- Bug in `roll_quantile` caused a memory leak when calling `.rolling(...).quantile(q)` with `q` in (0,1) (GH21965)

Missing

- Bug in *`Series.clip()`* and *`DataFrame.clip()`* cannot accept list-like threshold containing `NaN` (GH19992)

Contributors

A total of 6 people contributed patches to this release. People with a "+" by their names contributed a patch for the first time.

- Jeff Reback
- MeeseeksMachine +
- Tom Augspurger
- chris-b1
- h-vetinari
- meeseeksdev[bot]

5.4.2 What's new in 0.23.3 (July 7, 2018)

This release fixes a build issue with the sdist for Python 3.7 (GH21785) There are no other changes.

Contributors

A total of 2 people contributed patches to this release. People with a "+" by their names contributed a patch for the first time.

- Tom Augspurger
- meeseeksdev[bot] +

5.4.3 What's new in 0.23.2 (July 5, 2018)

This is a minor bug-fix release in the 0.23.x series and includes some small regression fixes and bug fixes. We recommend that all users upgrade to this version.

Note: Pandas 0.23.2 is first pandas release that's compatible with Python 3.7 (GH20552)

Warning: Starting January 1, 2019, pandas feature releases will support Python 3 only. See Dropping Python 2.7 for more.

What's new in v0.23.2

Logical reductions over entire DataFrame

DataFrame.all() and *DataFrame.any()* now accept `axis=None` to reduce over all axes to a scalar (GH19976)

```
In [1]: df = pd.DataFrame({"A": [1, 2], "B": [True, False]})

In [2]: df.all(axis=None)
Out[2]: False
```

This also provides compatibility with NumPy 1.15, which now dispatches to `DataFrame.all`. With NumPy 1.15 and pandas 0.23.1 or earlier, `numpy.all()` will no longer reduce over every axis:

```
>>> # NumPy 1.15, pandas 0.23.1
>>> np.any(pd.DataFrame({"A": [False], "B": [False]}))
A    False
B    False
dtype: bool
```

With pandas 0.23.2, that will correctly return False, as it did with NumPy < 1.15.

```
In [3]: np.any(pd.DataFrame({"A": [False], "B": [False]}))
Out[3]: False
```

Fixed regressions

- Fixed regression in `to_csv()` when handling file-like object incorrectly (GH21471)
- Re-allowed duplicate level names of a `MultiIndex`. Accessing a level that has a duplicate name by name still raises an error (GH19029).
- Bug in both *`DataFrame.first_valid_index()`* and *`Series.first_valid_index()`* raised for a row index having duplicate values (GH21441)
- Fixed printing of DataFrames with hierarchical columns with long names (GH21180)
- Fixed regression in *`reindex()`* and *`groupby()`* with a MultiIndex or multiple keys that contains categorical datetime-like values (GH21390).
- Fixed regression in unary negative operations with object dtype (GH21380)
- Bug in *`Timestamp.ceil()`* and *`Timestamp.floor()`* when timestamp is a multiple of the rounding frequency (GH21262)
- Fixed regression in `to_clipboard()` that defaulted to copying dataframes with space delimited instead of tab delimited (GH21104)

Build changes

- The source and binary distributions no longer include test data files, resulting in smaller download sizes. Tests relying on these data files will be skipped when using `pandas.test()`. (GH19320)

Bug fixes

Conversion

- Bug in constructing *`Index`* with an iterator or generator (GH21470)
- Bug in *`Series.nlargest()`* for signed and unsigned integer dtypes when the minimum value is present (GH21426)

Indexing

- Bug in *`Index.get_indexer_non_unique()`* with categorical key (GH21448)
- Bug in comparison operations for *`MultiIndex`* where error was raised on equality / inequality comparison involving a MultiIndex with `nlevels == 1` (GH21149)
- Bug in *`DataFrame.drop()`* behaviour is not consistent for unique and non-unique indexes (GH21494)
- Bug in *`DataFrame.duplicated()`* with a large number of columns causing a 'maximum recursion depth exceeded' (GH21524).

I/O

- Bug in *`read_csv()`* that caused it to incorrectly raise an error when `nrows=0`, `low_memory=True`, and `index_col` was not `None` (GH21141)
- Bug in *`json_normalize()`* when formatting the `record_prefix` with integer columns (GH21536)

Categorical

- Bug in rendering *`Series`* with `Categorical` dtype in rare conditions under Python 2.7 (GH21002)

Timezones

- Bug in *Timestamp* and *DatetimeIndex* where passing a *Timestamp* localized after a DST transition would return a datetime before the DST transition (GH20854)
- Bug in comparing *DataFrame* with tz-aware *DatetimeIndex* columns with a DST transition that raised a KeyError (GH19970)
- Bug in DatetimeIndex.shift() where an AssertionError would raise when shifting across DST (GH8616)
- Bug in *Timestamp* constructor where passing an invalid timezone offset designator (Z) would not raise a ValueError (GH8910)
- Bug in *Timestamp.replace()* where replacing at a DST boundary would retain an incorrect offset (GH7825)
- Bug in DatetimeIndex.reindex() when reindexing a tz-naive and tz-aware *DatetimeIndex* (GH8306)
- Bug in DatetimeIndex.resample() when downsampling across a DST boundary (GH8531)

Timedelta

- Bug in *Timedelta* where non-zero timedeltas shorter than 1 microsecond were considered False (GH21484)

Contributors

A total of 17 people contributed patches to this release. People with a "+" by their names contributed a patch for the first time.

- David Krych
- Jacopo Rota +
- Jeff Reback
- Jeremy Schendel
- Joris Van den Bossche
- Kalyan Gokhale
- Matthew Roeschke
- Michael Odintsov +
- Ming Li
- Pietro Battiston
- Tom Augspurger
- Uddeshya Singh
- Vu Le +
- alimcmaster1 +
- david-liu-brattle-1 +
- gfyoung
- jbrockmendel

5.4.4 What's new in 0.23.1 (June 12, 2018)

This is a minor bug-fix release in the 0.23.x series and includes some small regression fixes and bug fixes. We recommend that all users upgrade to this version.

Warning: Starting January 1, 2019, pandas feature releases will support Python 3 only. See Dropping Python 2.7 for more.

What's new in v0.23.1

Fixed regressions

Comparing Series with datetime.date

We've reverted a 0.23.0 change to comparing a *Series* holding datetimes and a `datetime.date` object (GH21152). In pandas 0.22 and earlier, comparing a Series holding datetimes and `datetime.date` objects would coerce the `datetime.date` to a datetime before comparing. This was inconsistent with Python, NumPy, and *DatetimeIndex*, which never consider a datetime and `datetime.date` equal.

In 0.23.0, we unified operations between DatetimeIndex and Series, and in the process changed comparisons between a Series of datetimes and `datetime.date` without warning.

We've temporarily restored the 0.22.0 behavior, so datetimes and dates may again compare equal, but restore the 0.23.0 behavior in a future release.

To summarize, here's the behavior in 0.22.0, 0.23.0, 0.23.1:

```
# 0.22.0... Silently coerce the datetime.date
>>> import datetime
>>> pd.Series(pd.date_range('2017', periods=2)) == datetime.date(2017, 1, 1)
0     True
1    False
dtype: bool

# 0.23.0... Do not coerce the datetime.date
>>> pd.Series(pd.date_range('2017', periods=2)) == datetime.date(2017, 1, 1)
0    False
1    False
dtype: bool

# 0.23.1... Coerce the datetime.date with a warning
>>> pd.Series(pd.date_range('2017', periods=2)) == datetime.date(2017, 1, 1)
/bin/python:1: FutureWarning: Comparing Series of datetimes with 'datetime.date'.
↪Currently, the
'datetime.date' is coerced to a datetime. In the future pandas will
not coerce, and the values not compare equal to the 'datetime.date'.
To retain the current behavior, convert the 'datetime.date' to a
```

(continues on next page)

(continued from previous page)

```
datetime with 'pd.Timestamp'.
  #!/bin/python3
0     True
1    False
dtype: bool
```

In addition, ordering comparisons will raise a `TypeError` in the future.

Other fixes

- Reverted the ability of *to_sql()* to perform multivalue inserts as this caused regression in certain cases (GH21103). In the future this will be made configurable.
- Fixed regression in the *DatetimeIndex.date* and *DatetimeIndex.time* attributes in case of timezone-aware data: *DatetimeIndex.time* returned a tz-aware time instead of tz-naive (GH21267) and *DatetimeIndex.date* returned incorrect date when the input date has a non-UTC timezone (GH21230).
- Fixed regression in `pandas.io.json.json_normalize()` when called with `None` values in nested levels in JSON, and to not drop keys with value as *None* (GH21158, GH21356).
- Bug in *to_csv()* causes encoding error when compression and encoding are specified (GH21241, GH21118)
- Bug preventing pandas from being importable with -OO optimization (GH21071)
- Bug in `Categorical.fillna()` incorrectly raising a `TypeError` when *value* the individual categories are iterable and *value* is an iterable (GH21097, GH19788)
- Fixed regression in constructors coercing NA values like `None` to strings when passing `dtype=str` (GH21083)
- Regression in *pivot_table()* where an ordered `Categorical` with missing values for the pivot's `index` would give a mis-aligned result (GH21133)
- Fixed regression in merging on boolean index/columns (GH21119).

Performance improvements

- Improved performance of `CategoricalIndex.is_monotonic_increasing()`, `CategoricalIndex.is_monotonic_decreasing()` and `CategoricalIndex.is_monotonic()` (GH21025)
- Improved performance of `CategoricalIndex.is_unique()` (GH21107)

Bug fixes

Groupby/resample/rolling

- Bug in *DataFrame.agg()* where applying multiple aggregation functions to a *DataFrame* with duplicated column names would cause a stack overflow (GH21063)
- Bug in *pandas.core.groupby.GroupBy.ffill()* and *pandas.core.groupby.GroupBy.bfill()* where the fill within a grouping would not always be applied as intended due to the implementations' use of a non-stable sort (GH21207)
- Bug in *pandas.core.groupby.GroupBy.rank()* where results did not scale to 100% when specifying `method='dense'` and `pct=True`
- Bug in *pandas.DataFrame.rolling()* and *pandas.Series.rolling()* which incorrectly accepted a 0 window size rather than raising (GH21286)

Data-type specific

- Bug in *Series.str.replace()* where the method throws *TypeError* on Python 3.5.2 (GH21078)
- Bug in *Timedelta* where passing a float with a unit would prematurely round the float precision (GH14156)
- Bug in *pandas.testing.assert_index_equal()* which raised `AssertionError` incorrectly, when comparing two *CategoricalIndex* objects with param `check_categorical=False` (GH19776)

Sparse

- Bug in `SparseArray.shape` which previously only returned the shape `SparseArray.sp_values` (GH21126)

Indexing

- Bug in *Series.reset_index()* where appropriate error was not raised with an invalid level name (GH20925)
- Bug in *interval_range()* when `start`/`periods` or `end`/`periods` are specified with float `start` or `end` (GH21161)
- Bug in `MultiIndex.set_names()` where error raised for a `MultiIndex` with `nlevels == 1` (GH21149)
- Bug in *IntervalIndex* constructors where creating an `IntervalIndex` from categorical data was not fully supported (GH21243, GH21253)
- Bug in `MultiIndex.sort_index()` which was not guaranteed to sort correctly with `level=1`; this was also causing data misalignment in particular *DataFrame.stack()* operations (GH20994, GH20945, GH21052)

Plotting

- New keywords (sharex, sharey) to turn on/off sharing of x/y-axis by subplots generated with pandas.DataFrame().groupby().boxplot() (GH20968)

I/O

- Bug in IO methods specifying `compression='zip'` which produced uncompressed zip archives (GH17778, GH21144)
- Bug in *DataFrame.to_stata()* which prevented exporting DataFrames to buffers and most file-like objects (GH21041)
- Bug in *read_stata()* and `StataReader` which did not correctly decode utf-8 strings on Python 3 from Stata 14 files (dta version 118) (GH21244)
- Bug in IO JSON *read_json()* reading empty JSON schema with `orient='table'` back to *DataFrame* caused an error (GH21287)

Reshaping

- Bug in *concat()* where error was raised in concatenating *Series* with numpy scalar and tuple names (GH21015)
- Bug in *concat()* warning message providing the wrong guidance for future behavior (GH21101)

Other

- Tab completion on *Index* in IPython no longer outputs deprecation warnings (GH21125)
- Bug preventing pandas being used on Windows without C++ redistributable installed (GH21106)

Contributors

A total of 30 people contributed patches to this release. People with a "+" by their names contributed a patch for the first time.

- Adam J. Stewart
- Adam Kim +
- Aly Sivji
- Chalmer Lowe +
- Damini Satya +
- Dr. Irv
- Gabe Fernando +
- Giftlin Rajaiah
- Jeff Reback
- Jeremy Schendel +
- Joris Van den Bossche
- Kalyan Gokhale +
- Kevin Sheppard
- Matthew Roeschke
- Max Kanter +
- Ming Li
- Pyry Kovanen +
- Stefano Cianciulli
- Tom Augspurger
- Uddeshya Singh +
- Wenhuan
- William Ayd
- chris-b1
- gfyoung
- h-vetinari
- nprad +
- ssikdar1 +
- tmnhat2001
- topper-123
- zertrin +

5.4.5 What's new in 0.23.0 (May 15, 2018)

This is a major release from 0.22.0 and includes a number of API changes, deprecations, new features, enhancements, and performance improvements along with a large number of bug fixes. We recommend that all users upgrade to this version.

Highlights include:

- *Round-trippable JSON format with 'table' orient*.
- *Instantiation from dicts respects order for Python 3.6+*.
- *Dependent column arguments for assign*.
- *Merging / sorting on a combination of columns and index levels*.
- *Extending pandas with custom types*.
- *Excluding unobserved categories from groupby*.
- *Changes to make output shape of DataFrame.apply consistent*.

Check the *API Changes* and *deprecations* before updating.

Warning: Starting January 1, 2019, pandas feature releases will support Python 3 only. See Dropping Python 2.7 for more.

What's new in v0.23.0

- *New features*
  - *JSON read/write round-trippable with* `orient='table'`
  - `.assign()` *accepts dependent arguments*
  - *Merging on a combination of columns and index levels*
  - *Sorting by a combination of columns and index levels*
  - *Extending pandas with custom types (experimental)*
  - *New* `observed` *keyword for excluding unobserved categories in* `groupby`
  - *Rolling/Expanding.apply() accepts* `raw=False` *to pass a* `Series` *to the function*
  - `DataFrame.interpolate` *has gained the* `limit_area` *kwarg*
  - `get_dummies` *now supports* `dtype` *argument*
  - *Timedelta mod method*
  - `.rank()` *handles* `inf` *values when* `NaN` *are present*
  - `Series.str.cat` *has gained the* `join` *kwarg*
  - `DataFrame.astype` *performs column-wise conversion to* `Categorical`
  - *Other enhancements*
- *Backwards incompatible API changes*
  - *Dependencies have increased minimum versions*
  - *Instantiation from dicts preserves dict insertion order for python 3.6+*

New features

JSON read/write round-trippable with `orient='table'`

A `DataFrame` can now be written to and subsequently read back via JSON while preserving metadata through usage of the `orient='table'` argument (see GH18912 and GH9146). Previously, none of the available `orient` values guaranteed the preservation of dtypes and index names, amongst other metadata.

```
In [1]: df = pd.DataFrame({'foo': [1, 2, 3, 4],
   ...:                    'bar': ['a', 'b', 'c', 'd'],
   ...:                    'baz': pd.date_range('2018-01-01', freq='d', periods=4),
   ...:                    'qux': pd.Categorical(['a', 'b', 'c', 'c'])},
   ...:                   index=pd.Index(range(4), name='idx'))
   ...:

In [2]: df
Out[2]:
     foo bar        baz qux
idx
0      1   a 2018-01-01   a
1      2   b 2018-01-02   b
2      3   c 2018-01-03   c
3      4   d 2018-01-04   c

[4 rows x 4 columns]

In [3]: df.dtypes
Out[3]:
foo              int64
bar             object
baz    datetime64[ns]
qux          category
Length: 4, dtype: object

In [4]: df.to_json('test.json', orient='table')

In [5]: new_df = pd.read_json('test.json', orient='table')

In [6]: new_df
Out[6]:
     foo bar        baz qux
idx
0      1   a 2018-01-01   a
1      2   b 2018-01-02   b
2      3   c 2018-01-03   c
3      4   d 2018-01-04   c

[4 rows x 4 columns]

In [7]: new_df.dtypes
Out[7]:
foo              int64
bar             object
baz    datetime64[ns]
qux          category
Length: 4, dtype: object
```

Please note that the string *index* is not supported with the round trip format, as it is used by default in `write_json`

to indicate a missing index name.

```
In [8]: df.index.name = 'index'

In [9]: df.to_json('test.json', orient='table')

In [10]: new_df = pd.read_json('test.json', orient='table')

In [11]: new_df
Out[11]:
   foo bar        baz qux
0    1   a 2018-01-01   a
1    2   b 2018-01-02   b
2    3   c 2018-01-03   c
3    4   d 2018-01-04   c

[4 rows x 4 columns]

In [12]: new_df.dtypes
Out[12]:
foo              int64
bar             object
baz     datetime64[ns]
qux           category
Length: 4, dtype: object
```

`.assign()` accepts dependent arguments

The *`DataFrame.assign()`* now accepts dependent keyword arguments for python version later than 3.6 (see also PEP 468). Later keyword arguments may now refer to earlier ones if the argument is a callable. See the *documentation here* (GH14207)

```
In [13]: df = pd.DataFrame({'A': [1, 2, 3]})

In [14]: df
Out[14]:
   A
0  1
1  2
2  3

[3 rows x 1 columns]

In [15]: df.assign(B=df.A, C=lambda x: x['A'] + x['B'])
Out[15]:
   A  B  C
0  1  1  2
1  2  2  4
2  3  3  6

[3 rows x 3 columns]
```

Warning: This may subtly change the behavior of your code when you're using `.assign()` to update an existing column. Previously, callables referring to other variables being updated would get the "old" values

Previous behavior:

```
In [2]: df = pd.DataFrame({"A": [1, 2, 3]})

In [3]: df.assign(A=lambda df: df.A + 1, C=lambda df: df.A * -1)
Out[3]:
   A  C
0  2 -1
1  3 -2
2  4 -3
```

New behavior:

```
In [16]: df.assign(A=df.A + 1, C=lambda df: df.A * -1)
Out[16]:
   A  C
0  2 -2
1  3 -3
2  4 -4

[3 rows x 2 columns]
```

Merging on a combination of columns and index levels

Strings passed to *DataFrame.merge()* as the `on`, `left_on`, and `right_on` parameters may now refer to either column names or index level names. This enables merging `DataFrame` instances on a combination of index levels and columns without resetting indexes. See the *Merge on columns and levels* documentation section. (GH14355)

```
In [17]: left_index = pd.Index(['K0', 'K0', 'K1', 'K2'], name='key1')

In [18]: left = pd.DataFrame({'A': ['A0', 'A1', 'A2', 'A3'],
   ....:                      'B': ['B0', 'B1', 'B2', 'B3'],
   ....:                      'key2': ['K0', 'K1', 'K0', 'K1']},
   ....:                     index=left_index)
   ....:

In [19]: right_index = pd.Index(['K0', 'K1', 'K2', 'K2'], name='key1')

In [20]: right = pd.DataFrame({'C': ['C0', 'C1', 'C2', 'C3'],
   ....:                       'D': ['D0', 'D1', 'D2', 'D3'],
   ....:                       'key2': ['K0', 'K0', 'K0', 'K1']},
   ....:                      index=right_index)
   ....:

In [21]: left.merge(right, on=['key1', 'key2'])
Out[21]:
       A   B key2   C   D
key1
K0    A0  B0   K0  C0  D0
K1    A2  B2   K0  C1  D1
K2    A3  B3   K1  C3  D3

[3 rows x 5 columns]
```

Sorting by a combination of columns and index levels

Strings passed to *DataFrame.sort_values()* as the `by` parameter may now refer to either column names or index level names. This enables sorting `DataFrame` instances by a combination of index levels and columns without resetting indexes. See the *Sorting by Indexes and Values* documentation section. (GH14353)

```
# Build MultiIndex
In [22]: idx = pd.MultiIndex.from_tuples([('a', 1), ('a', 2), ('a', 2),
   ....:                                  ('b', 2), ('b', 1), ('b', 1)])
   ....:

In [23]: idx.names = ['first', 'second']

# Build DataFrame
In [24]: df_multi = pd.DataFrame({'A': np.arange(6, 0, -1)},
   ....:                         index=idx)
   ....:

In [25]: df_multi
Out[25]:
              A
first second
a     1       6
      2       5
      2       4
b     2       3
      1       2
      1       1

[6 rows x 1 columns]

# Sort by 'second' (index) and 'A' (column)
In [26]: df_multi.sort_values(by=['second', 'A'])
Out[26]:
              A
first second
b     1       1
      1       2
a     1       6
b     2       3
a     2       4
      2       5

[6 rows x 1 columns]
```

Extending pandas with custom types (experimental)

Pandas now supports storing array-like objects that aren't necessarily 1-D NumPy arrays as columns in a DataFrame or values in a Series. This allows third-party libraries to implement extensions to NumPy's types, similar to how pandas implemented categoricals, datetimes with timezones, periods, and intervals.

As a demonstration, we'll use cyberpandas, which provides an `IPArray` type for storing ip addresses.

```
In [1]: from cyberpandas import IPArray

In [2]: values = IPArray([
```

(continues on next page)

(continued from previous page)

```
   ...:     0,
   ...:     3232235777,
   ...:     42540766452641154071740215577757643572
   ...: ])
   ...:
   ...:
```

`IPArray` isn't a normal 1-D NumPy array, but because it's a pandas *`ExtensionArray`*, it can be stored properly inside pandas' containers.

```
In [3]: ser = pd.Series(values)

In [4]: ser
Out[4]:
0                         0.0.0.0
1                     192.168.1.1
2    2001:db8:85a3::8a2e:370:7334
dtype: ip
```

Notice that the dtype is `ip`. The missing value semantics of the underlying array are respected:

```
In [5]: ser.isna()
Out[5]:
0     True
1    False
2    False
dtype: bool
```

For more, see the *extension types* documentation. If you build an extension array, publicize it on our ecosystem page.

New `observed` keyword for excluding unobserved categories in `groupby`

Grouping by a categorical includes the unobserved categories in the output. When grouping by multiple categorical columns, this means you get the cartesian product of all the categories, including combinations where there are no observations, which can result in a large number of groups. We have added a keyword `observed` to control this behavior, it defaults to `observed=False` for backward-compatibility. (GH14942, GH8138, GH15217, GH17594, GH8669, GH20583, GH20902)

```
In [27]: cat1 = pd.Categorical(["a", "a", "b", "b"],
   ....:                       categories=["a", "b", "z"], ordered=True)
   ....:

In [28]: cat2 = pd.Categorical(["c", "d", "c", "d"],
   ....:                       categories=["c", "d", "y"], ordered=True)
   ....:

In [29]: df = pd.DataFrame({"A": cat1, "B": cat2, "values": [1, 2, 3, 4]})

In [30]: df['C'] = ['foo', 'bar'] * 2

In [31]: df
Out[31]:
   A  B  values    C
0  a  c       1  foo
1  a  d       2  bar
```

(continues on next page)

(continued from previous page)

```
2  b  c       3  foo
3  b  d       4  bar

[4 rows x 4 columns]
```

To show all values, the previous behavior:

```
In [32]: df.groupby(['A', 'B', 'C'], observed=False).count()
Out[32]:
         values
A B C
a c bar     NaN
    foo     1.0
  d bar     1.0
    foo     NaN
  y bar     NaN
...         ...
z c foo     NaN
  d bar     NaN
    foo     NaN
  y bar     NaN
    foo     NaN

[18 rows x 1 columns]
```

To show only observed values:

```
In [33]: df.groupby(['A', 'B', 'C'], observed=True).count()
Out[33]:
         values
A B C
a c foo       1
  d bar       1
b c foo       1
  d bar       1

[4 rows x 1 columns]
```

For pivoting operations, this behavior is *already* controlled by the `dropna` keyword:

```
In [34]: cat1 = pd.Categorical(["a", "a", "b", "b"],
   ....:                       categories=["a", "b", "z"], ordered=True)
   ....:

In [35]: cat2 = pd.Categorical(["c", "d", "c", "d"],
   ....:                       categories=["c", "d", "y"], ordered=True)
   ....:

In [36]: df = pd.DataFrame({"A": cat1, "B": cat2, "values": [1, 2, 3, 4]})

In [37]: df
Out[37]:
   A  B  values
0  a  c       1
1  a  d       2
2  b  c       3
3  b  d       4
```

(continues on next page)

(continued from previous page)

```
[4 rows x 3 columns]
```

```
In [38]: pd.pivot_table(df, values='values', index=['A', 'B'],
   ....:                  dropna=True)
   ....:
Out[38]:
     values
A B
a c       1
  d       2
b c       3
  d       4

[4 rows x 1 columns]

In [39]: pd.pivot_table(df, values='values', index=['A', 'B'],
   ....:                  dropna=False)
   ....:
Out[39]:
     values
A B
a c     1.0
  d     2.0
  y     NaN
b c     3.0
  d     4.0
  y     NaN
z c     NaN
  d     NaN
  y     NaN

[9 rows x 1 columns]
```

Rolling/Expanding.apply() accepts `raw=False` to pass a `Series` to the function

`Series.rolling().apply()`, `DataFrame.rolling().apply()`, `Series.expanding().apply()`, and `DataFrame.expanding().apply()` have gained a `raw=None` parameter. This is similar to `DataFame.apply()`. This parameter, if `True` allows one to send a `np.ndarray` to the applied function. If `False` a `Series` will be passed. The default is `None`, which preserves backward compatibility, so this will default to `True`, sending an `np.ndarray`. In a future version the default will be changed to `False`, sending a `Series`. (GH5071, GH20584)

```
In [40]: s = pd.Series(np.arange(5), np.arange(5) + 1)

In [41]: s
Out[41]:
1    0
2    1
3    2
4    3
5    4
Length: 5, dtype: int64
```

Pass a `Series`: