io.excel.xlsm.reader [string] The default Excel reader engine for 'xlsm' files. Available options: auto, xlrd, openpyxl. [default: auto] [currently: auto]

io.excel.xlsm.writer [string] The default Excel writer engine for 'xlsm' files. Available options: auto, openpyxl. [default: auto] [currently: auto]

io.excel.xlsx.reader [string] The default Excel reader engine for 'xlsx' files. Available options: auto, xlrd, openpyxl. [default: auto] [currently: auto]

io.excel.xlsx.writer [string] The default Excel writer engine for 'xlsx' files. Available options: auto, openpyxl, xlsxwriter. [default: auto] [currently: auto]

io.hdf.default_format [format] default format writing format, if None, then put will default to 'fixed' and append will default to 'table' [default: None] [currently: None]

io.hdf.dropna_table [boolean] drop ALL nan rows when appending to a table [default: False] [currently: False]

io.parquet.engine [string] The default parquet reader/writer engine. Available options: 'auto', 'pyarrow', 'fastparquet', the default is 'auto' [default: auto] [currently: auto]

mode.chained_assignment [string] Raise an exception, warn, or no action if trying to use chained assignment, The default is warn [default: warn] [currently: warn]

mode.sim_interactive [boolean] Whether to simulate interactive mode for purposes of testing [default: False] [currently: False]

mode.use_inf_as_na [boolean] True means treat None, NaN, INF, -INF as NA (old way), False means None and NaN are null, but INF, -INF are not NA (new way). [default: False] [currently: False]

mode.use_inf_as_null [boolean] use_inf_as_null had been deprecated and will be removed in a future version. Use *use_inf_as_na* instead. [default: False] [currently: False] (Deprecated, use *mode.use_inf_as_na* instead.)

plotting.backend [str] The plotting backend to use. The default value is "matplotlib", the backend provided with pandas. Other backends can be specified by prodiving the name of the module that implements the backend. [default: matplotlib] [currently: matplotlib]

plotting.matplotlib.register_converters [bool or 'auto'.] Whether to register converters with matplotlib's units registry for dates, times, datetimes, and Periods. Toggling to False will remove the converters, restoring any converters that pandas overwrote. [default: auto] [currently: auto]

pandas.reset_option

pandas.**reset_option**(*pat*) **= <pandas._config.config.CallableDynamicDoc object>**

Reset one or more options to their default value.

Pass "all" as argument to reset all options.

Available options:

- compute.[use_bottleneck, use_numexpr]
- display.[chop_threshold, colheader_justify, column_space, date_dayfirst, date_yearfirst, encoding, expand_frame_repr, float_format]
- display.html.[border, table_schema, use_mathjax]
- display.[large_repr]
- display.latex.[escape, longtable, multicolumn, multicolumn_format, multirow, repr]
- display.[max_categories, max_columns, max_colwidth, max_info_columns, max_info_rows, max_rows, max_seq_items, memory_usage, min_rows, multi_sparse, notebook_repr_html, pprint_nest_depth, precision, show_dimensions]
- display.unicode.[ambiguous_as_wide, east_asian_width]
- display.[width]
- io.excel.ods.[reader]
- io.excel.xls.[reader, writer]
- io.excel.xlsb.[reader]
- io.excel.xlsm.[reader, writer]

- io.excel.xlsx.[reader, writer]
- io.hdf.[default_format, dropna_table]
- io.parquet.[engine]
- mode.[chained_assignment, sim_interactive, use_inf_as_na, use_inf_as_null]
- plotting.[backend]
- plotting.matplotlib.[register_converters]

Parameters

pat [str/regex] If specified only options matching *prefix** will be reset. Note: partial matches are supported for convenience, but unless you use the full option name (e.g. x.y.z.option_name), your code may break in future versions if new options with similar names are introduced.

Returns

None

Notes

The available options with its descriptions:

compute.use_bottleneck [bool] Use the bottleneck library to accelerate if it is installed, the default is True Valid values: False,True [default: True] [currently: True]

compute.use_numexpr [bool] Use the numexpr library to accelerate computation if it is installed, the default is True Valid values: False,True [default: True] [currently: True]

display.chop_threshold [float or None] if set to a float value, all float values smaller then the given threshold will be displayed as exactly 0 by repr and friends. [default: None] [currently: None]

display.colheader_justify ['left'/'right'] Controls the justification of column headers. used by DataFrameFormatter. [default: right] [currently: right]

display.column_space No description available. [default: 12] [currently: 12]

display.date_dayfirst [boolean] When True, prints and parses dates with the day first, eg 20/01/2005 [default: False] [currently: False]

display.date_yearfirst [boolean] When True, prints and parses dates with the year first, eg 2005/01/20 [default: False] [currently: False]

display.encoding [str/unicode] Defaults to the detected encoding of the console. Specifies the encoding to be used for strings returned by to_string, these are generally strings meant to be displayed on the console. [default: UTF-8] [currently: UTF-8]

display.expand_frame_repr [boolean] Whether to print out the full DataFrame repr for wide DataFrames across multiple lines, *max_columns* is still respected, but the output will wrap-around across multiple "pages" if its width exceeds *display.width*. [default: True] [currently: True]

display.float_format [callable] The callable should accept a floating point number and return a string with the desired format of the number. This is used in some places like SeriesFormatter. See formats.format.EngFormatter for an example. [default: None] [currently: None]

display.html.border [int] A `border=value` attribute is inserted in the `<table>` tag for the DataFrame HTML repr. [default: 1] [currently: 1]

display.html.table_schema [boolean] Whether to publish a Table Schema representation for frontends that support it. (default: False) [default: False] [currently: False]

display.html.use_mathjax [boolean] When True, Jupyter notebook will process table contents using MathJax, rendering mathematical expressions enclosed by the dollar symbol. (default: True) [default: True] [currently: True]

display.large_repr ['truncate'/'info'] For DataFrames exceeding max_rows/max_cols, the repr (and HTML repr) can show a truncated table (the default from 0.13), or switch to the view from df.info() (the behaviour in earlier versions of pandas). [default: truncate] [currently: truncate]

display.latex.escape [bool] This specifies if the to_latex method of a Dataframe uses escapes special characters. Valid values: False,True [default: True] [currently: True]

display.latex.longtable :bool This specifies if the to_latex method of a Dataframe uses the longtable format. Valid values: False,True [default: False] [currently: False]

display.latex.multicolumn [bool] This specifies if the to_latex method of a Dataframe uses multicolumns to pretty-print MultiIndex columns. Valid values: False,True [default: True] [currently: True]

display.latex.multicolumn_format [bool] This specifies if the to_latex method of a Dataframe uses multicolumns to pretty-print MultiIndex columns. Valid values: False,True [default: l] [currently: l]

display.latex.multirow [bool] This specifies if the to_latex method of a Dataframe uses multirows to pretty-print MultiIndex rows. Valid values: False,True [default: False] [currently: False]

display.latex.repr [boolean] Whether to produce a latex DataFrame representation for jupyter environments that support it. (default: False) [default: False] [currently: False]

display.max_categories [int] This sets the maximum number of categories pandas should output when printing out a *Categorical* or a Series of dtype "category". [default: 8] [currently: 8]

display.max_columns [int] If max_cols is exceeded, switch to truncate view. Depending on *large_repr*, objects are either centrally truncated or printed as a summary view. 'None' value means unlimited.

In case python/IPython is running in a terminal and *large_repr* equals 'truncate' this can be set to 0 and pandas will auto-detect the width of the terminal and print a truncated object which fits the screen width. The IPython notebook, IPython qtconsole, or IDLE do not run in a terminal and hence it is not possible to do correct auto-detection. [default: 0] [currently: 0]

display.max_colwidth [int or None] The maximum width in characters of a column in the repr of a pandas data structure. When the column overflows, a "..." placeholder is embedded in the output. A 'None' value means unlimited. [default: 50] [currently: 50]

display.max_info_columns [int] max_info_columns is used in DataFrame.info method to decide if per column information will be printed. [default: 100] [currently: 100]

display.max_info_rows [int or None] df.info() will usually show null-counts for each column. For large frames this can be quite slow. max_info_rows and max_info_cols limit this null check only to frames with smaller dimensions than specified. [default: 1690785] [currently: 1690785]

display.max_rows [int] If max_rows is exceeded, switch to truncate view. Depending on *large_repr*, objects are either centrally truncated or printed as a summary view. 'None' value means unlimited.

In case python/IPython is running in a terminal and *large_repr* equals 'truncate' this can be set to 0 and pandas will auto-detect the height of the terminal and print a truncated object which fits the screen height. The IPython notebook, IPython qtconsole, or IDLE do not run in a terminal and hence it is not possible to do correct auto-detection. [default: 60] [currently: 15]

display.max_seq_items [int or None] when pretty-printing a long sequence, no more then *max_seq_items* will be printed. If items are omitted, they will be denoted by the addition of "..." to the resulting string.

If set to None, the number of items to be printed is unlimited. [default: 100] [currently: 100]

display.memory_usage [bool, string or None] This specifies if the memory usage of a DataFrame should be displayed when df.info() is called. Valid values True,False,'deep' [default: True] [currently: True]

display.min_rows [int] The numbers of rows to show in a truncated view (when *max_rows* is exceeded). Ignored when *max_rows* is set to None or 0. When set to None, follows the value of *max_rows*. [default: 10] [currently: 10]

display.multi_sparse [boolean] "sparsify" MultiIndex display (don't display repeated elements in outer levels within groups) [default: True] [currently: True]

display.notebook_repr_html [boolean] When True, IPython notebook will use html representation for pandas objects (if it is available). [default: True] [currently: True]

display.pprint_nest_depth [int] Controls the number of nested levels to process when pretty-printing [default: 3] [currently: 3]

display.precision [int] Floating point output precision (number of significant digits). This is only a suggestion [default: 6] [currently: 6]

display.show_dimensions [boolean or 'truncate'] Whether to print out dimensions at the end of DataFrame repr. If 'truncate' is specified, only print out the dimensions if the frame is truncated (e.g. not display all rows and/or columns) [default: truncate] [currently: truncate]

display.unicode.ambiguous_as_wide [boolean] Whether to use the Unicode East Asian Width to calculate the

display text width. Enabling this may affect to the performance (default: False) [default: False] [currently: False]

display.unicode.east_asian_width [boolean] Whether to use the Unicode East Asian Width to calculate the display text width. Enabling this may affect to the performance (default: False) [default: False] [currently: False]

display.width [int] Width of the display in characters. In case python/IPython is running in a terminal this can be set to None and pandas will correctly auto-detect the width. Note that the IPython notebook, IPython qtconsole, or IDLE do not run in a terminal and hence it is not possible to correctly detect the width. [default: 80] [currently: 80]

io.excel.ods.reader [string] The default Excel reader engine for 'ods' files. Available options: auto, odf. [default: auto] [currently: auto]

io.excel.xls.reader [string] The default Excel reader engine for 'xls' files. Available options: auto, xlrd. [default: auto] [currently: auto]

io.excel.xls.writer [string] The default Excel writer engine for 'xls' files. Available options: auto, xlwt. [default: auto] [currently: auto]

io.excel.xlsb.reader [string] The default Excel reader engine for 'xlsb' files. Available options: auto, pyxlsb. [default: auto] [currently: auto]

io.excel.xlsm.reader [string] The default Excel reader engine for 'xlsm' files. Available options: auto, xlrd, openpyxl. [default: auto] [currently: auto]

io.excel.xlsm.writer [string] The default Excel writer engine for 'xlsm' files. Available options: auto, openpyxl. [default: auto] [currently: auto]

io.excel.xlsx.reader [string] The default Excel reader engine for 'xlsx' files. Available options: auto, xlrd, openpyxl. [default: auto] [currently: auto]

io.excel.xlsx.writer [string] The default Excel writer engine for 'xlsx' files. Available options: auto, openpyxl, xlsxwriter. [default: auto] [currently: auto]

io.hdf.default_format [format] default format writing format, if None, then put will default to 'fixed' and append will default to 'table' [default: None] [currently: None]

io.hdf.dropna_table [boolean] drop ALL nan rows when appending to a table [default: False] [currently: False]

io.parquet.engine [string] The default parquet reader/writer engine. Available options: 'auto', 'pyarrow', 'fastparquet', the default is 'auto' [default: auto] [currently: auto]

mode.chained_assignment [string] Raise an exception, warn, or no action if trying to use chained assignment, The default is warn [default: warn] [currently: warn]

mode.sim_interactive [boolean] Whether to simulate interactive mode for purposes of testing [default: False] [currently: False]

mode.use_inf_as_na [boolean] True means treat None, NaN, INF, -INF as NA (old way), False means None and NaN are null, but INF, -INF are not NA (new way). [default: False] [currently: False]

mode.use_inf_as_null [boolean] use_inf_as_null had been deprecated and will be removed in a future version. Use *use_inf_as_na* instead. [default: False] [currently: False] (Deprecated, use *mode.use_inf_as_na* instead.)

plotting.backend [str] The plotting backend to use. The default value is "matplotlib", the backend provided with pandas. Other backends can be specified by prodiving the name of the module that implements the backend. [default: matplotlib] [currently: matplotlib]

plotting.matplotlib.register_converters [bool or 'auto'.] Whether to register converters with matplotlib's units registry for dates, times, datetimes, and Periods. Toggling to False will remove the converters, restoring any converters that pandas overwrote. [default: auto] [currently: auto]

pandas.get_option

pandas.**get_option**(*pat*) **= <pandas._config.config.CallableDynamicDoc object>**

Retrieves the value of the specified option.

Available options:

- compute.[use_bottleneck, use_numexpr]
- display.[chop_threshold, colheader_justify, column_space, date_dayfirst, date_yearfirst, encoding, expand_frame_repr, float_format]
- display.html.[border, table_schema, use_mathjax]
- display.[large_repr]
- display.latex.[escape, longtable, multicolumn, multicolumn_format, multirow, repr]
- display.[max_categories, max_columns, max_colwidth, max_info_columns, max_info_rows, max_rows, max_seq_items, memory_usage, min_rows, multi_sparse, notebook_repr_html, pprint_nest_depth, precision, show_dimensions]
- display.unicode.[ambiguous_as_wide, east_asian_width]
- display.[width]
- io.excel.ods.[reader]
- io.excel.xls.[reader, writer]
- io.excel.xlsb.[reader]
- io.excel.xlsm.[reader, writer]
- io.excel.xlsx.[reader, writer]
- io.hdf.[default_format, dropna_table]
- io.parquet.[engine]
- mode.[chained_assignment, sim_interactive, use_inf_as_na, use_inf_as_null]
- plotting.[backend]
- plotting.matplotlib.[register_converters]

Parameters

pat [str] Regexp which should match a single option. Note: partial matches are supported for convenience, but unless you use the full option name (e.g. x.y.z.option_name), your code may break in future versions if new options with similar names are introduced.

Returns

result [the value of the option]

Raises

OptionError [if no such option exists]

Notes

The available options with its descriptions:

compute.use_bottleneck [bool] Use the bottleneck library to accelerate if it is installed, the default is True Valid values: False,True [default: True] [currently: True]

compute.use_numexpr [bool] Use the numexpr library to accelerate computation if it is installed, the default is True Valid values: False,True [default: True] [currently: True]

display.chop_threshold [float or None] if set to a float value, all float values smaller then the given threshold will be displayed as exactly 0 by repr and friends. [default: None] [currently: None]

display.colheader_justify ['left'/'right'] Controls the justification of column headers. used by DataFrameFormatter. [default: right] [currently: right]

display.column_space No description available. [default: 12] [currently: 12]

display.date_dayfirst [boolean] When True, prints and parses dates with the day first, eg 20/01/2005 [default: False] [currently: False]

display.date_yearfirst [boolean] When True, prints and parses dates with the year first, eg 2005/01/20 [default: False] [currently: False]

display.encoding [str/unicode] Defaults to the detected encoding of the console. Specifies the encoding to be used for strings returned by to_string, these are generally strings meant to be displayed on the console. [default: UTF-8] [currently: UTF-8]

display.expand_frame_repr [boolean] Whether to print out the full DataFrame repr for wide DataFrames across multiple lines, *max_columns* is still respected, but the output will wrap-around across multiple "pages" if its width exceeds *display.width*. [default: True] [currently: True]

display.float_format [callable] The callable should accept a floating point number and return a string with the desired format of the number. This is used in some places like SeriesFormatter. See formats.format.EngFormatter for an example. [default: None] [currently: None]

display.html.border [int] A `border=value` attribute is inserted in the `<table>` tag for the DataFrame HTML repr. [default: 1] [currently: 1]

display.html.table_schema [boolean] Whether to publish a Table Schema representation for frontends that support it. (default: False) [default: False] [currently: False]

display.html.use_mathjax [boolean] When True, Jupyter notebook will process table contents using MathJax, rendering mathematical expressions enclosed by the dollar symbol. (default: True) [default: True] [currently: True]

display.large_repr ['truncate'/'info'] For DataFrames exceeding max_rows/max_cols, the repr (and HTML repr) can show a truncated table (the default from 0.13), or switch to the view from df.info() (the behaviour in earlier versions of pandas). [default: truncate] [currently: truncate]

display.latex.escape [bool] This specifies if the to_latex method of a Dataframe uses escapes special characters. Valid values: False,True [default: True] [currently: True]

display.latex.longtable :bool This specifies if the to_latex method of a Dataframe uses the longtable format. Valid values: False,True [default: False] [currently: False]

display.latex.multicolumn [bool] This specifies if the to_latex method of a Dataframe uses multicolumns to pretty-print MultiIndex columns. Valid values: False,True [default: True] [currently: True]

display.latex.multicolumn_format [bool] This specifies if the to_latex method of a Dataframe uses multicolumns to pretty-print MultiIndex columns. Valid values: False,True [default: l] [currently: l]

display.latex.multirow [bool] This specifies if the to_latex method of a Dataframe uses multirows to pretty-print MultiIndex rows. Valid values: False,True [default: False] [currently: False]

display.latex.repr [boolean] Whether to produce a latex DataFrame representation for jupyter environments that support it. (default: False) [default: False] [currently: False]

display.max_categories [int] This sets the maximum number of categories pandas should output when printing out a *Categorical* or a Series of dtype "category". [default: 8] [currently: 8]

display.max_columns [int] If max_cols is exceeded, switch to truncate view. Depending on *large_repr*, objects are either centrally truncated or printed as a summary view. 'None' value means unlimited.

In case python/IPython is running in a terminal and *large_repr* equals 'truncate' this can be set to 0 and pandas will auto-detect the width of the terminal and print a truncated object which fits the screen width. The IPython notebook, IPython qtconsole, or IDLE do not run in a terminal and hence it is not possible to do correct auto-detection. [default: 0] [currently: 0]

display.max_colwidth [int or None] The maximum width in characters of a column in the repr of a pandas data structure. When the column overflows, a "..." placeholder is embedded in the output. A 'None' value means unlimited. [default: 50] [currently: 50]

display.max_info_columns [int] max_info_columns is used in DataFrame.info method to decide if per column information will be printed. [default: 100] [currently: 100]

display.max_info_rows [int or None] df.info() will usually show null-counts for each column. For large frames this can be quite slow. max_info_rows and max_info_cols limit this null check only to frames with smaller dimensions than specified. [default: 1690785] [currently: 1690785]

display.max_rows [int] If max_rows is exceeded, switch to truncate view. Depending on *large_repr*, objects are either centrally truncated or printed as a summary view. 'None' value means unlimited.

In case python/IPython is running in a terminal and *large_repr* equals 'truncate' this can be set to 0 and pandas will auto-detect the height of the terminal and print a truncated object which fits the screen height.

The IPython notebook, IPython qtconsole, or IDLE do not run in a terminal and hence it is not possible to do correct auto-detection. [default: 60] [currently: 15]

display.max_seq_items [int or None] when pretty-printing a long sequence, no more then *max_seq_items* will be printed. If items are omitted, they will be denoted by the addition of "..." to the resulting string.

If set to None, the number of items to be printed is unlimited. [default: 100] [currently: 100]

display.memory_usage [bool, string or None] This specifies if the memory usage of a DataFrame should be displayed when df.info() is called. Valid values True,False,'deep' [default: True] [currently: True]

display.min_rows [int] The numbers of rows to show in a truncated view (when *max_rows* is exceeded). Ignored when *max_rows* is set to None or 0. When set to None, follows the value of *max_rows*. [default: 10] [currently: 10]

display.multi_sparse [boolean] "sparsify" MultiIndex display (don't display repeated elements in outer levels within groups) [default: True] [currently: True]

display.notebook_repr_html [boolean] When True, IPython notebook will use html representation for pandas objects (if it is available). [default: True] [currently: True]

display.pprint_nest_depth [int] Controls the number of nested levels to process when pretty-printing [default: 3] [currently: 3]

display.precision [int] Floating point output precision (number of significant digits). This is only a suggestion [default: 6] [currently: 6]

display.show_dimensions [boolean or 'truncate'] Whether to print out dimensions at the end of DataFrame repr. If 'truncate' is specified, only print out the dimensions if the frame is truncated (e.g. not display all rows and/or columns) [default: truncate] [currently: truncate]

display.unicode.ambiguous_as_wide [boolean] Whether to use the Unicode East Asian Width to calculate the display text width. Enabling this may affect to the performance (default: False) [default: False] [currently: False]

display.unicode.east_asian_width [boolean] Whether to use the Unicode East Asian Width to calculate the display text width. Enabling this may affect to the performance (default: False) [default: False] [currently: False]

display.width [int] Width of the display in characters. In case python/IPython is running in a terminal this can be set to None and pandas will correctly auto-detect the width. Note that the IPython notebook, IPython qtconsole, or IDLE do not run in a terminal and hence it is not possible to correctly detect the width. [default: 80] [currently: 80]

io.excel.ods.reader [string] The default Excel reader engine for 'ods' files. Available options: auto, odf. [default: auto] [currently: auto]

io.excel.xls.reader [string] The default Excel reader engine for 'xls' files. Available options: auto, xlrd. [default: auto] [currently: auto]

io.excel.xls.writer [string] The default Excel writer engine for 'xls' files. Available options: auto, xlwt. [default: auto] [currently: auto]

io.excel.xlsb.reader [string] The default Excel reader engine for 'xlsb' files. Available options: auto, pyxlsb. [default: auto] [currently: auto]

io.excel.xlsm.reader [string] The default Excel reader engine for 'xlsm' files. Available options: auto, xlrd, openpyxl. [default: auto] [currently: auto]

io.excel.xlsm.writer [string] The default Excel writer engine for 'xlsm' files. Available options: auto, openpyxl. [default: auto] [currently: auto]

io.excel.xlsx.reader [string] The default Excel reader engine for 'xlsx' files. Available options: auto, xlrd, openpyxl. [default: auto] [currently: auto]

io.excel.xlsx.writer [string] The default Excel writer engine for 'xlsx' files. Available options: auto, openpyxl, xlsxwriter. [default: auto] [currently: auto]

io.hdf.default_format [format] default format writing format, if None, then put will default to 'fixed' and append will default to 'table' [default: None] [currently: None]

io.hdf.dropna_table [boolean] drop ALL nan rows when appending to a table [default: False] [currently: False]

io.parquet.engine [string] The default parquet reader/writer engine. Available options: 'auto', 'pyarrow', 'fastparquet', the default is 'auto' [default: auto] [currently: auto]

mode.chained_assignment [string] Raise an exception, warn, or no action if trying to use chained assignment, The default is warn [default: warn] [currently: warn]

mode.sim_interactive [boolean] Whether to simulate interactive mode for purposes of testing [default: False] [currently: False]

mode.use_inf_as_na [boolean] True means treat None, NaN, INF, -INF as NA (old way), False means None and NaN are null, but INF, -INF are not NA (new way). [default: False] [currently: False]

mode.use_inf_as_null [boolean] use_inf_as_null had been deprecated and will be removed in a future version. Use *use_inf_as_na* instead. [default: False] [currently: False] (Deprecated, use *mode.use_inf_as_na* instead.)

plotting.backend [str] The plotting backend to use. The default value is "matplotlib", the backend provided with pandas. Other backends can be specified by prodiving the name of the module that implements the backend. [default: matplotlib] [currently: matplotlib]

plotting.matplotlib.register_converters [bool or 'auto'.] Whether to register converters with matplotlib's units registry for dates, times, datetimes, and Periods. Toggling to False will remove the converters, restoring any converters that pandas overwrote. [default: auto] [currently: auto]

pandas.set_option

pandas.**set_option**(*pat*, *value*) **= <pandas._config.config.CallableDynamicDoc object>**

Sets the value of the specified option.

Available options:

- compute.[use_bottleneck, use_numexpr]
- display.[chop_threshold, colheader_justify, column_space, date_dayfirst, date_yearfirst, encoding, expand_frame_repr, float_format]
- display.html.[border, table_schema, use_mathjax]
- display.[large_repr]
- display.latex.[escape, longtable, multicolumn, multicolumn_format, multirow, repr]
- display.[max_categories, max_columns, max_colwidth, max_info_columns, max_info_rows, max_rows, max_seq_items, memory_usage, min_rows, multi_sparse, notebook_repr_html, pprint_nest_depth, precision, show_dimensions]
- display.unicode.[ambiguous_as_wide, east_asian_width]
- display.[width]
- io.excel.ods.[reader]
- io.excel.xls.[reader, writer]
- io.excel.xlsb.[reader]
- io.excel.xlsm.[reader, writer]
- io.excel.xlsx.[reader, writer]
- io.hdf.[default_format, dropna_table]
- io.parquet.[engine]
- mode.[chained_assignment, sim_interactive, use_inf_as_na, use_inf_as_null]
- plotting.[backend]
- plotting.matplotlib.[register_converters]

Parameters

pat [str] Regexp which should match a single option. Note: partial matches are supported for convenience, but unless you use the full option name (e.g. x.y.z.option_name), your code may break in future versions if new options with similar names are introduced.

value [object] New value of option.

Returns

None

Raises

OptionError if no such option exists

Notes

The available options with its descriptions:

compute.use_bottleneck [bool] Use the bottleneck library to accelerate if it is installed, the default is True Valid values: False,True [default: True] [currently: True]

compute.use_numexpr [bool] Use the numexpr library to accelerate computation if it is installed, the default is True Valid values: False,True [default: True] [currently: True]

display.chop_threshold [float or None] if set to a float value, all float values smaller then the given threshold will be displayed as exactly 0 by repr and friends. [default: None] [currently: None]

display.colheader_justify ['left'/'right'] Controls the justification of column headers. used by DataFrameFormatter. [default: right] [currently: right]

display.column_space No description available. [default: 12] [currently: 12]

display.date_dayfirst [boolean] When True, prints and parses dates with the day first, eg 20/01/2005 [default: False] [currently: False]

display.date_yearfirst [boolean] When True, prints and parses dates with the year first, eg 2005/01/20 [default: False] [currently: False]

display.encoding [str/unicode] Defaults to the detected encoding of the console. Specifies the encoding to be used for strings returned by to_string, these are generally strings meant to be displayed on the console. [default: UTF-8] [currently: UTF-8]

display.expand_frame_repr [boolean] Whether to print out the full DataFrame repr for wide DataFrames across multiple lines, *max_columns* is still respected, but the output will wrap-around across multiple "pages" if its width exceeds *display.width*. [default: True] [currently: True]

display.float_format [callable] The callable should accept a floating point number and return a string with the desired format of the number. This is used in some places like SeriesFormatter. See formats.format.EngFormatter for an example. [default: None] [currently: None]

display.html.border [int] A `border=value` attribute is inserted in the `<table>` tag for the DataFrame HTML repr. [default: 1] [currently: 1]

display.html.table_schema [boolean] Whether to publish a Table Schema representation for frontends that support it. (default: False) [default: False] [currently: False]

display.html.use_mathjax [boolean] When True, Jupyter notebook will process table contents using MathJax, rendering mathematical expressions enclosed by the dollar symbol. (default: True) [default: True] [currently: True]

display.large_repr ['truncate'/'info'] For DataFrames exceeding max_rows/max_cols, the repr (and HTML repr) can show a truncated table (the default from 0.13), or switch to the view from df.info() (the behaviour in earlier versions of pandas). [default: truncate] [currently: truncate]

display.latex.escape [bool] This specifies if the to_latex method of a Dataframe uses escapes special characters. Valid values: False,True [default: True] [currently: True]

display.latex.longtable :bool This specifies if the to_latex method of a Dataframe uses the longtable format. Valid values: False,True [default: False] [currently: False]

display.latex.multicolumn [bool] This specifies if the to_latex method of a Dataframe uses multicolumns to pretty-print MultiIndex columns. Valid values: False,True [default: True] [currently: True]

display.latex.multicolumn_format [bool] This specifies if the to_latex method of a Dataframe uses multicolumns to pretty-print MultiIndex columns. Valid values: False,True [default: l] [currently: l]

display.latex.multirow [bool] This specifies if the to_latex method of a Dataframe uses multirows to pretty-print MultiIndex rows. Valid values: False,True [default: False] [currently: False]

display.latex.repr [boolean] Whether to produce a latex DataFrame representation for jupyter environments that support it. (default: False) [default: False] [currently: False]

display.max_categories [int] This sets the maximum number of categories pandas should output when printing out a *Categorical* or a Series of dtype "category". [default: 8] [currently: 8]

display.max_columns [int] If max_cols is exceeded, switch to truncate view. Depending on *large_repr*, objects are either centrally truncated or printed as a summary view. 'None' value means unlimited.

In case python/IPython is running in a terminal and *large_repr* equals 'truncate' this can be set to 0 and pandas will auto-detect the width of the terminal and print a truncated object which fits the screen width. The IPython notebook, IPython qtconsole, or IDLE do not run in a terminal and hence it is not possible to do correct auto-detection. [default: 0] [currently: 0]

display.max_colwidth [int or None] The maximum width in characters of a column in the repr of a pandas data structure. When the column overflows, a "..." placeholder is embedded in the output. A 'None' value means unlimited. [default: 50] [currently: 50]

display.max_info_columns [int] max_info_columns is used in DataFrame.info method to decide if per column information will be printed. [default: 100] [currently: 100]

display.max_info_rows [int or None] df.info() will usually show null-counts for each column. For large frames this can be quite slow. max_info_rows and max_info_cols limit this null check only to frames with smaller dimensions than specified. [default: 1690785] [currently: 1690785]

display.max_rows [int] If max_rows is exceeded, switch to truncate view. Depending on *large_repr*, objects are either centrally truncated or printed as a summary view. 'None' value means unlimited.

In case python/IPython is running in a terminal and *large_repr* equals 'truncate' this can be set to 0 and pandas will auto-detect the height of the terminal and print a truncated object which fits the screen height. The IPython notebook, IPython qtconsole, or IDLE do not run in a terminal and hence it is not possible to do correct auto-detection. [default: 60] [currently: 15]

display.max_seq_items [int or None] when pretty-printing a long sequence, no more then *max_seq_items* will be printed. If items are omitted, they will be denoted by the addition of "..." to the resulting string.

If set to None, the number of items to be printed is unlimited. [default: 100] [currently: 100]

display.memory_usage [bool, string or None] This specifies if the memory usage of a DataFrame should be displayed when df.info() is called. Valid values True,False,'deep' [default: True] [currently: True]

display.min_rows [int] The numbers of rows to show in a truncated view (when *max_rows* is exceeded). Ignored when *max_rows* is set to None or 0. When set to None, follows the value of *max_rows*. [default: 10] [currently: 10]

display.multi_sparse [boolean] "sparsify" MultiIndex display (don't display repeated elements in outer levels within groups) [default: True] [currently: True]

display.notebook_repr_html [boolean] When True, IPython notebook will use html representation for pandas objects (if it is available). [default: True] [currently: True]

display.pprint_nest_depth [int] Controls the number of nested levels to process when pretty-printing [default: 3] [currently: 3]

display.precision [int] Floating point output precision (number of significant digits). This is only a suggestion [default: 6] [currently: 6]

display.show_dimensions [boolean or 'truncate'] Whether to print out dimensions at the end of DataFrame repr. If 'truncate' is specified, only print out the dimensions if the frame is truncated (e.g. not display all rows and/or columns) [default: truncate] [currently: truncate]

display.unicode.ambiguous_as_wide [boolean] Whether to use the Unicode East Asian Width to calculate the display text width. Enabling this may affect to the performance (default: False) [default: False] [currently: False]

display.unicode.east_asian_width [boolean] Whether to use the Unicode East Asian Width to calculate the display text width. Enabling this may affect to the performance (default: False) [default: False] [currently: False]

display.width [int] Width of the display in characters. In case python/IPython is running in a terminal this can be set to None and pandas will correctly auto-detect the width. Note that the IPython notebook, IPython qtconsole, or IDLE do not run in a terminal and hence it is not possible to correctly detect the width. [default: 80] [currently: 80]

io.excel.ods.reader [string] The default Excel reader engine for 'ods' files. Available options: auto, odf. [default: auto] [currently: auto]

io.excel.xls.reader [string] The default Excel reader engine for 'xls' files. Available options: auto, xlrd. [de-

fault: auto] [currently: auto]

io.excel.xls.writer [string] The default Excel writer engine for 'xls' files. Available options: auto, xlwt. [default: auto] [currently: auto]

io.excel.xlsb.reader [string] The default Excel reader engine for 'xlsb' files. Available options: auto, pyxlsb. [default: auto] [currently: auto]

io.excel.xlsm.reader [string] The default Excel reader engine for 'xlsm' files. Available options: auto, xlrd, openpyxl. [default: auto] [currently: auto]

io.excel.xlsm.writer [string] The default Excel writer engine for 'xlsm' files. Available options: auto, openpyxl. [default: auto] [currently: auto]

io.excel.xlsx.reader [string] The default Excel reader engine for 'xlsx' files. Available options: auto, xlrd, openpyxl. [default: auto] [currently: auto]

io.excel.xlsx.writer [string] The default Excel writer engine for 'xlsx' files. Available options: auto, openpyxl, xlsxwriter. [default: auto] [currently: auto]

io.hdf.default_format [format] default format writing format, if None, then put will default to 'fixed' and append will default to 'table' [default: None] [currently: None]

io.hdf.dropna_table [boolean] drop ALL nan rows when appending to a table [default: False] [currently: False]

io.parquet.engine [string] The default parquet reader/writer engine. Available options: 'auto', 'pyarrow', 'fastparquet', the default is 'auto' [default: auto] [currently: auto]

mode.chained_assignment [string] Raise an exception, warn, or no action if trying to use chained assignment, The default is warn [default: warn] [currently: warn]

mode.sim_interactive [boolean] Whether to simulate interactive mode for purposes of testing [default: False] [currently: False]

mode.use_inf_as_na [boolean] True means treat None, NaN, INF, -INF as NA (old way), False means None and NaN are null, but INF, -INF are not NA (new way). [default: False] [currently: False]

mode.use_inf_as_null [boolean] use_inf_as_null had been deprecated and will be removed in a future version. Use *use_inf_as_na* instead. [default: False] [currently: False] (Deprecated, use *mode.use_inf_as_na* instead.)

plotting.backend [str] The plotting backend to use. The default value is "matplotlib", the backend provided with pandas. Other backends can be specified by prodiving the name of the module that implements the backend. [default: matplotlib] [currently: matplotlib]

plotting.matplotlib.register_converters [bool or 'auto'.] Whether to register converters with matplotlib's units registry for dates, times, datetimes, and Periods. Toggling to False will remove the converters, restoring any converters that pandas overwrote. [default: auto] [currently: auto]

pandas.option_context

class pandas.**option_context**(**args*)

Context manager to temporarily set options in the *with* statement context.

You need to invoke as `option_context(pat, val, [(pat, val), ...])`.

Examples

```
>>> with option_context('display.max_rows', 10, 'display.max_columns', 5):
...     ...
```

3.15.2 Testing functions

testing.assert_frame_equal(left, right[, ...])	Check that left and right DataFrame are equal.
testing.assert_series_equal(left, right[, ...])	Check that left and right Series are equal.
testing.assert_index_equal(left, right, ...)	Check that left and right Index are equal.
testing.assert_extension_array_equal(left, right)	Check that left and right ExtensionArrays are equal.

pandas.testing.assert_frame_equal

pandas.testing.**assert_frame_equal**(*left*, *right*, *check_dtype=True*, *check_index_type='equiv'*, *check_column_type='equiv'*, *check_frame_type=True*, *check_less_precise=False*, *check_names=True*, *by_blocks=False*, *check_exact=False*, *check_datetimelike_compat=False*, *check_categorical=True*, *check_like=False*, *obj='DataFrame'*)

Check that left and right DataFrame are equal.

This function is intended to compare two DataFrames and output any differences. Is is mostly intended for use in unit tests. Additional parameters allow varying the strictness of the equality checks performed.

Parameters

left [DataFrame] First DataFrame to compare.

right [DataFrame] Second DataFrame to compare.

check_dtype [bool, default True] Whether to check the DataFrame dtype is identical.

check_index_type [bool or {'equiv'}, default 'equiv'] Whether to check the Index class, dtype and inferred_type are identical.

check_column_type [bool or {'equiv'}, default 'equiv'] Whether to check the columns class, dtype and inferred_type are identical. Is passed as the `exact` argument of *assert_index_equal()*.

check_frame_type [bool, default True] Whether to check the DataFrame class is identical.

check_less_precise [bool or int, default False] Specify comparison precision. Only used when check_exact is False. 5 digits (False) or 3 digits (True) after decimal points are compared. If int, then specify the digits to compare.

When comparing two numbers, if the first number has magnitude less than 1e-5, we compare the two numbers directly and check whether they are equivalent within the specified precision. Otherwise, we compare the **ratio** of the second number to the first number and check whether it is equivalent to 1 within the specified precision.

check_names [bool, default True] Whether to check that the *names* attribute for both the *index* and *column* attributes of the DataFrame is identical.

by_blocks [bool, default False] Specify how to compare internal data. If False, compare by columns. If True, compare by blocks.

check_exact [bool, default False] Whether to compare number exactly.

check_datetimelike_compat [bool, default False] Compare datetime-like which is comparable ignoring dtype.

check_categorical [bool, default True] Whether to compare internal Categorical exactly.

check_like [bool, default False] If True, ignore the order of index & columns. Note: index labels must match their respective rows (same as in columns) - same labels must be with the same data.

obj [str, default 'DataFrame'] Specify object name being compared, internally used to show appropriate assertion message.

See also:

assert_series_equal Equivalent method for asserting Series equality.
DataFrame.equals Check DataFrame equality.

Examples

This example shows comparing two DataFrames that are equal but with columns of differing dtypes.

```
>>> from pandas._testing import assert_frame_equal
>>> df1 = pd.DataFrame({'a': [1, 2], 'b': [3, 4]})
>>> df2 = pd.DataFrame({'a': [1, 2], 'b': [3.0, 4.0]})
```

df1 equals itself.

```
>>> assert_frame_equal(df1, df1)
```

df1 differs from df2 as column 'b' is of a different type.

```
>>> assert_frame_equal(df1, df2)
Traceback (most recent call last):
...
AssertionError: Attributes of DataFrame.iloc[:, 1] (column name="b") are different
```

Attribute "dtype" are different [left]: int64 [right]: float64

Ignore differing dtypes in columns with check_dtype.

```
>>> assert_frame_equal(df1, df2, check_dtype=False)
```

pandas.testing.assert_series_equal

pandas.testing.**assert_series_equal**(*left*, *right*, *check_dtype=True*, *check_index_type='equiv'*, *check_series_type=True*, *check_less_precise=False*, *check_names=True*, *check_exact=False*, *check_datetimelike_compat=False*, *check_categorical=True*, *check_category_order=True*, *obj='Series'*)

Check that left and right Series are equal.

Parameters

left [Series]

right [Series]

check_dtype [bool, default True] Whether to check the Series dtype is identical.

check_index_type [bool or {'equiv'}, default 'equiv'] Whether to check the Index class, dtype and inferred_type are identical.

check_series_type [bool, default True] Whether to check the Series class is identical.

check_less_precise [bool or int, default False] Specify comparison precision. Only used when check_exact is False. 5 digits (False) or 3 digits (True) after decimal points are compared. If int, then specify the digits to compare.

When comparing two numbers, if the first number has magnitude less than 1e-5, we compare the two numbers directly and check whether they are equivalent within the specified precision. Otherwise, we compare the **ratio** of the second number to the first number and check whether it is equivalent to 1 within the specified precision.

check_names [bool, default True] Whether to check the Series and Index names attribute.

check_exact [bool, default False] Whether to compare number exactly.

check_datetimelike_compat [bool, default False] Compare datetime-like which is comparable ignoring dtype.

check_categorical [bool, default True] Whether to compare internal Categorical exactly.

check_category_order [bool, default True] Whether to compare category order of internal Categoricals

New in version 1.0.2.

obj [str, default 'Series'] Specify object name being compared, internally used to show appropriate assertion message.

pandas.testing.assert_index_equal

pandas.testing.**assert_index_equal**(*left: pandas.core.indexes.base.Index*, *right: pandas.core.indexes.base.Index*, *exact: Union[bool, str] = 'equiv'*, *check_names: bool = True*, *check_less_precise: Union[bool, int] = False*, *check_exact: bool = True*, *check_categorical: bool = True*, *obj: str = 'Index'*) → None

Check that left and right Index are equal.

Parameters

left [Index]

right [Index]

exact [bool or {'equiv'}, default 'equiv'] Whether to check the Index class, dtype and inferred_type are identical. If 'equiv', then RangeIndex can be substituted for Int64Index as well.

check_names [bool, default True] Whether to check the names attribute.

check_less_precise [bool or int, default False] Specify comparison precision. Only used when check_exact is False. 5 digits (False) or 3 digits (True) after decimal points are compared. If int, then specify the digits to compare.

check_exact [bool, default True] Whether to compare number exactly.

check_categorical [bool, default True] Whether to compare internal Categorical exactly.

obj [str, default 'Index'] Specify object name being compared, internally used to show appropriate assertion message.

pandas.testing.assert_extension_array_equal

pandas.testing.**assert_extension_array_equal**(*left*, *right*, *check_dtype=True*, *check_less_precise=False*, *check_exact=False*)

Check that left and right ExtensionArrays are equal.

Parameters

left, right [ExtensionArray] The two arrays to compare

check_dtype [bool, default True] Whether to check if the ExtensionArray dtypes are identical.

check_less_precise [bool or int, default False] Specify comparison precision. Only used when check_exact is False. 5 digits (False) or 3 digits (True) after decimal points are compared. If int, then specify the digits to compare.

check_exact [bool, default False] Whether to compare number exactly.

Notes

Missing values are checked separately from valid values. A mask of missing values is computed for each and checked to match. The remaining all-valid values are cast to object dtype and checked.

3.15.3 Exceptions and warnings

errors.DtypeWarning	Warning raised when reading different dtypes in a column from a file.
errors.EmptyDataError	Exception that is thrown in *pd.read_csv* (by both the C and Python engines) when empty data or header is encountered.
errors.OutOfBoundsDatetime	
errors.ParserError	Exception that is raised by an error encountered in parsing file contents.
errors.ParserWarning	Warning raised when reading a file that doesn't use the default 'c' parser.
errors.PerformanceWarning	Warning raised when there is a possible performance impact.
errors.UnsortedIndexError	Error raised when attempting to get a slice of a MultiIndex, and the index has not been lexsorted.
errors.UnsupportedFunctionCall	Exception raised when attempting to call a numpy function on a pandas object, but that function is not supported by the object e.g.

pandas.errors.DtypeWarning

exception pandas.errors.**DtypeWarning**

Warning raised when reading different dtypes in a column from a file.

Raised for a dtype incompatibility. This can happen whenever *read_csv* or *read_table* encounter non-uniform dtypes in a column(s) of a given CSV file.

See also:

read_csv Read CSV (comma-separated) file into a DataFrame.
read_table Read general delimited file into a DataFrame.

Notes

This warning is issued when dealing with larger files because the dtype checking happens per chunk read.

Despite the warning, the CSV file is read with mixed types in a single column which will be an object type. See the examples below to better understand this issue.

Examples

This example creates and reads a large CSV file with a column that contains *int* and *str*.

```
>>> df = pd.DataFrame({'a': (['1'] * 100000 + ['X'] * 100000 +
...                          ['1'] * 100000),
...                    'b': ['b'] * 300000})
>>> df.to_csv('test.csv', index=False)
>>> df2 = pd.read_csv('test.csv')
... # DtypeWarning: Columns (0) have mixed types
```

Important to notice that `df2` will contain both *str* and *int* for the same input, '1'.

```
>>> df2.iloc[262140, 0]
'1'
>>> type(df2.iloc[262140, 0])
<class 'str'>
>>> df2.iloc[262150, 0]
1
>>> type(df2.iloc[262150, 0])
<class 'int'>
```

One way to solve this issue is using the *dtype* parameter in the *read_csv* and *read_table* functions to explicit the conversion:

```
>>> df2 = pd.read_csv('test.csv', sep=',', dtype={'a': str})
```

No warning was issued.

```
>>> import os
>>> os.remove('test.csv')
```

pandas.errors.EmptyDataError

exception pandas.errors.**EmptyDataError**
Exception that is thrown in *pd.read_csv* (by both the C and Python engines) when empty data or header is encountered.

pandas.errors.OutOfBoundsDatetime

exception pandas.errors.**OutOfBoundsDatetime**

pandas.errors.ParserError

exception pandas.errors.**ParserError**
Exception that is raised by an error encountered in parsing file contents.

This is a generic error raised for errors encountered when functions like *read_csv* or *read_html* are parsing contents of a file.

See also:

read_csv Read CSV (comma-separated) file into a DataFrame.
read_html Read HTML table into a DataFrame.

pandas.errors.ParserWarning

exception pandas.errors.**ParserWarning**
Warning raised when reading a file that doesn't use the default 'c' parser.

Raised by *pd.read_csv* and *pd.read_table* when it is necessary to change parsers, generally from the default 'c' parser to 'python'.

It happens due to a lack of support or functionality for parsing a particular attribute of a CSV file with the requested engine.

Currently, 'c' unsupported options include the following parameters:

1. *sep* other than a single character (e.g. regex separators)
2. *skipfooter* higher than 0
3. *sep=None* with *delim_whitespace=False*

The warning can be avoided by adding *engine='python'* as a parameter in *pd.read_csv* and *pd.read_table* methods.

See also:

pd.read_csv Read CSV (comma-separated) file into DataFrame.
pd.read_table Read general delimited file into DataFrame.

Examples

Using a *sep* in *pd.read_csv* other than a single character:

```
>>> import io
>>> csv = '''a;b;c
...           1;1,8
...           1;2,1'''
>>> df = pd.read_csv(io.StringIO(csv), sep='[;,]')
... # ParserWarning: Falling back to the 'python' engine...
```

Adding *engine='python'* to *pd.read_csv* removes the Warning:

```
>>> df = pd.read_csv(io.StringIO(csv), sep='[;,]', engine='python')
```

pandas.errors.PerformanceWarning

exception pandas.errors.**PerformanceWarning**
Warning raised when there is a possible performance impact.

pandas.errors.UnsortedIndexError

exception pandas.errors.**UnsortedIndexError**
Error raised when attempting to get a slice of a MultiIndex, and the index has not been lexsorted. Subclass of *KeyError*.

pandas.errors.UnsupportedFunctionCall

exception pandas.errors.**UnsupportedFunctionCall**
Exception raised when attempting to call a numpy function on a pandas object, but that function is not supported by the object e.g. `np.cumsum(groupby_object)`.

3.15.4 Data types related functionality

api.types.union_categoricals(to_union, ...)	Combine list-like of Categorical-like, unioning categories.
api.types.infer_dtype()	Efficiently infer the type of a passed val, or list-like array of values.
api.types.pandas_dtype(dtype)	Convert input into a pandas only dtype object or a numpy dtype object.

pandas.api.types.union_categoricals

pandas.api.types.**union_categoricals**(*to_union*, *sort_categories: bool = False*, *ignore_order: bool = False*)

Combine list-like of Categorical-like, unioning categories.

All categories must have the same dtype.

Parameters

to_union [list-like] Categorical, CategoricalIndex, or Series with dtype='category'.

sort_categories [bool, default False] If true, resulting categories will be lexsorted, otherwise they will be ordered as they appear in the data.

ignore_order [bool, default False] If true, the ordered attribute of the Categoricals will be ignored. Results in an unordered categorical.

Returns

Categorical

Raises

TypeError

- all inputs do not have the same dtype
- all inputs do not have the same ordered property
- all inputs are ordered and their categories are not identical
- sort_categories=True and Categoricals are ordered

ValueError Empty list of categoricals passed

Notes

To learn more about categories, see link

Examples

```
>>> from pandas.api.types import union_categoricals
```

If you want to combine categoricals that do not necessarily have the same categories, *union_categoricals* will combine a list-like of categoricals. The new categories will be the union of the categories being combined.

```
>>> a = pd.Categorical(["b", "c"])
>>> b = pd.Categorical(["a", "b"])
>>> union_categoricals([a, b])
[b, c, a, b]
Categories (3, object): [b, c, a]
```

By default, the resulting categories will be ordered as they appear in the *categories* of the data. If you want the categories to be lexsorted, use *sort_categories=True* argument.

```
>>> union_categoricals([a, b], sort_categories=True)
[b, c, a, b]
Categories (3, object): [a, b, c]
```

union_categoricals also works with the case of combining two categoricals of the same categories and order information (e.g. what you could also *append* for).

```
>>> a = pd.Categorical(["a", "b"], ordered=True)
>>> b = pd.Categorical(["a", "b", "a"], ordered=True)
>>> union_categoricals([a, b])
[a, b, a, b, a]
Categories (2, object): [a < b]
```

Raises *TypeError* because the categories are ordered and not identical.

```
>>> a = pd.Categorical(["a", "b"], ordered=True)
>>> b = pd.Categorical(["a", "b", "c"], ordered=True)
>>> union_categoricals([a, b])
TypeError: to union ordered Categoricals, all categories must be the same
```

New in version 0.20.0

Ordered categoricals with different categories or orderings can be combined by using the *ignore_ordered=True* argument.

```
>>> a = pd.Categorical(["a", "b", "c"], ordered=True)
>>> b = pd.Categorical(["c", "b", "a"], ordered=True)
>>> union_categoricals([a, b], ignore_order=True)
[a, b, c, c, b, a]
Categories (3, object): [a, b, c]
```

union_categoricals also works with a *CategoricalIndex*, or *Series* containing categorical data, but note that the resulting array will always be a plain *Categorical*

```
>>> a = pd.Series(["b", "c"], dtype='category')
>>> b = pd.Series(["a", "b"], dtype='category')
>>> union_categoricals([a, b])
[b, c, a, b]
Categories (3, object): [b, c, a]
```

pandas.api.types.infer_dtype

pandas.api.types.**infer_dtype**()

Efficiently infer the type of a passed val, or list-like array of values. Return a string describing the type.

Parameters

value [scalar, list, ndarray, or pandas type]

skipna [bool, default True] Ignore NaN values when inferring the type.

New in version 0.21.0.

Returns

str Describing the common type of the input data.

Results can include:

- **string**
- **bytes**
- **floating**

- **integer**
- **mixed-integer**
- **mixed-integer-float**
- **decimal**
- **complex**
- **categorical**
- **boolean**
- **datetime64**
- **datetime**
- **date**
- **timedelta64**
- **timedelta**
- **time**
- **period**
- **mixed**

Raises

TypeError If ndarray-like but cannot infer the dtype

Notes

- 'mixed' is the catchall for anything that is not otherwise specialized
- 'mixed-integer-float' are floats and integers
- 'mixed-integer' are integers mixed with non-integers

Examples

```
>>> infer_dtype(['foo', 'bar'])
'string'
```

```
>>> infer_dtype(['a', np.nan, 'b'], skipna=True)
'string'
```

```
>>> infer_dtype(['a', np.nan, 'b'], skipna=False)
'mixed'
```

```
>>> infer_dtype([b'foo', b'bar'])
'bytes'
```

```
>>> infer_dtype([1, 2, 3])
'integer'
```

```
>>> infer_dtype([1, 2, 3.5])
'mixed-integer-float'
```

```
>>> infer_dtype([1.0, 2.0, 3.5])
'floating'
```

```
>>> infer_dtype(['a', 1])
'mixed-integer'
```

```
>>> infer_dtype([Decimal(1), Decimal(2.0)])
'decimal'
```

```
>>> infer_dtype([True, False])
'boolean'
```

```
>>> infer_dtype([True, False, np.nan])
'mixed'
```

```
>>> infer_dtype([pd.Timestamp('20130101')])
'datetime'
```

```
>>> infer_dtype([datetime.date(2013, 1, 1)])
'date'
```

```
>>> infer_dtype([np.datetime64('2013-01-01')])
'datetime64'
```

```
>>> infer_dtype([datetime.timedelta(0, 1, 1)])
'timedelta'
```

```
>>> infer_dtype(pd.Series(list('aabc')).astype('category'))
'categorical'
```

pandas.api.types.pandas_dtype

pandas.api.types.**pandas_dtype**(*dtype*)

Convert input into a pandas only dtype object or a numpy dtype object.

Parameters

dtype [object to be converted]

Returns

np.dtype or a pandas dtype

Raises

TypeError if not a dtype

Dtype introspection

api.types.is_bool_dtype(arr_or_dtype)	Check whether the provided array or dtype is of a boolean dtype.
api.types.is_categorical_dtype(arr_or_dtype)	Check whether an array-like or dtype is of the Categorical dtype.
api.types.is_complex_dtype(arr_or_dtype)	Check whether the provided array or dtype is of a complex dtype.
api.types.is_datetime64_any_dtype(arr_or_dtype)	Check whether the provided array or dtype is of the datetime64 dtype.
api.types.is_datetime64_dtype(arr_or_dtype)	Check whether an array-like or dtype is of the datetime64 dtype.
api.types.is_datetime64_ns_dtype(arr_or_dtype)	Check whether the provided array or dtype is of the datetime64[ns] dtype.
api.types.is_datetime64tz_dtype(arr_or_dtype)	Check whether an array-like or dtype is of a DatetimeTZDtype dtype.
api.types.is_extension_type(arr)	(DEPRECATED) Check whether an array-like is of a pandas extension class instance.
api.types.is_extension_array_dtype(arr_or_dtype)	Check if an object is a pandas extension array type.
api.types.is_float_dtype(arr_or_dtype)	Check whether the provided array or dtype is of a float dtype.
api.types.is_int64_dtype(arr_or_dtype)	Check whether the provided array or dtype is of the int64 dtype.
api.types.is_integer_dtype(arr_or_dtype)	Check whether the provided array or dtype is of an integer dtype.
api.types.is_interval_dtype(arr_or_dtype)	Check whether an array-like or dtype is of the Interval dtype.
api.types.is_numeric_dtype(arr_or_dtype)	Check whether the provided array or dtype is of a numeric dtype.
api.types.is_object_dtype(arr_or_dtype)	Check whether an array-like or dtype is of the object dtype.
api.types.is_period_dtype(arr_or_dtype)	Check whether an array-like or dtype is of the Period dtype.
api.types.is_signed_integer_dtype(arr_or_dtype)	Check whether the provided array or dtype is of a signed integer dtype.
api.types.is_string_dtype(arr_or_dtype)	Check whether the provided array or dtype is of the string dtype.
api.types.is_timedelta64_dtype(arr_or_dtype)	Check whether an array-like or dtype is of the timedelta64 dtype.
api.types.is_timedelta64_ns_dtype(arr_or_dtype)	Check whether the provided array or dtype is of the timedelta64[ns] dtype.
api.types.is_unsigned_integer_dtype(arr_or_dtype)	Check whether the provided array or dtype is of an unsigned integer dtype.
api.types.is_sparse(arr)	Check whether an array-like is a 1-D pandas sparse array.

pandas.api.types.is_bool_dtype

pandas.api.types.**is_bool_dtype**(*arr_or_dtype*) → bool

Check whether the provided array or dtype is of a boolean dtype.

Parameters

arr_or_dtype [array-like] The array or dtype to check.

Returns

boolean Whether or not the array or dtype is of a boolean dtype.

Notes

An ExtensionArray is considered boolean when the `_is_boolean` attribute is set to True.

Examples

```
>>> is_bool_dtype(str)
False
>>> is_bool_dtype(int)
False
>>> is_bool_dtype(bool)
True
>>> is_bool_dtype(np.bool)
True
>>> is_bool_dtype(np.array(['a', 'b']))
False
>>> is_bool_dtype(pd.Series([1, 2]))
False
>>> is_bool_dtype(np.array([True, False]))
True
>>> is_bool_dtype(pd.Categorical([True, False]))
True
>>> is_bool_dtype(pd.arrays.SparseArray([True, False]))
True
```

pandas.api.types.is_categorical_dtype

pandas.api.types.**is_categorical_dtype**(*arr_or_dtype*) → bool

Check whether an array-like or dtype is of the Categorical dtype.

Parameters

arr_or_dtype [array-like] The array-like or dtype to check.

Returns

boolean Whether or not the array-like or dtype is of the Categorical dtype.

Examples

```
>>> is_categorical_dtype(object)
False
>>> is_categorical_dtype(CategoricalDtype())
True
>>> is_categorical_dtype([1, 2, 3])
False
>>> is_categorical_dtype(pd.Categorical([1, 2, 3]))
True
>>> is_categorical_dtype(pd.CategoricalIndex([1, 2, 3]))
True
```

pandas.api.types.is_complex_dtype

pandas.api.types.**is_complex_dtype**(*arr_or_dtype*) → bool

Check whether the provided array or dtype is of a complex dtype.

Parameters

arr_or_dtype [array-like] The array or dtype to check.

Returns

boolean Whether or not the array or dtype is of a complex dtype.

Examples

```
>>> is_complex_dtype(str)
False
>>> is_complex_dtype(int)
False
>>> is_complex_dtype(np.complex)
True
>>> is_complex_dtype(np.array(['a', 'b']))
False
>>> is_complex_dtype(pd.Series([1, 2]))
False
>>> is_complex_dtype(np.array([1 + 1j, 5]))
True
```

pandas.api.types.is_datetime64_any_dtype

pandas.api.types.**is_datetime64_any_dtype**(*arr_or_dtype*) → bool

Check whether the provided array or dtype is of the datetime64 dtype.

Parameters

arr_or_dtype [array-like] The array or dtype to check.

Returns

boolean Whether or not the array or dtype is of the datetime64 dtype.