Tip

This page only contains information on the `st.data_editor` API. For an overview of working with dataframes and to learn more about the data editor's capabilities and limitations, read Dataframes.

st.data_editor

Streamlit Version Version 1.41.0

Display a data editor widget.

The data editor widget allows you to edit dataframes and many other data structures in a table-like UI.

Function signature[source]

st.data_editor(data, *, width=None, height=None, use_container_width=False, hide_index=None, column_order=None, column_config=None, num_rows="fixed", disabled=False, key=None, on_change=None, args=None, kwargs=None)

Parameters

data (Anything supported by st.dataframe)	The data to edit in the data editor. Note • Styles from `pandas.Styler` will only be applied to non-editable columns. • Text and number formatting from `column_config` always takes precedence over text and number formatting from `pandas.Styler`. • Mixing data types within a column can make the column uneditable. • Additionally, the following data types are not yet supported for editing: `complex`, `list`, `tuple`, `bytes`, `bytearray`, `memoryview`, `dict`, `set`, `frozenset`, `fractions.Fraction`, `pandas.Interval`, and `pandas.Period`. • To prevent overflow in JavaScript, columns containing `datetime.timedelta` and `pandas.Timedelta` values will default to uneditable, but this can be changed through column configuration.
width (int or None)	Desired width of the data editor expressed in pixels. If `width` is `None` (default), Streamlit sets the data editor width to fit its contents up to the width of the parent container. If `width` is greater than the width of the parent container, Streamlit sets the data editor width to match the width of the parent container.
height (int or None)	Desired height of the data editor expressed in pixels. If `height` is `None` (default), Streamlit sets the height to show at most ten rows. Vertical scrolling within the data editor element is enabled when the height does not accomodate all rows.
use_container_width (bool)	Whether to override `width` with the width of the parent container. If `use_container_width` is `False` (default), Streamlit sets the data editor's width according to `width`. If `use_container_width` is `True`, Streamlit sets the width of the data editor to match the width of the parent container.
hide_index (bool or None)	Whether to hide the index column(s). If `hide_index` is `None` (default), the visibility of index columns is automatically determined based on the data.

Returns

(pandas.DataFrame, pandas.Series, pyarrow.Table, numpy.ndarray, list, set, tuple, or dict.)	The edited data. The edited data is returned in its original data type if it corresponds to any of the supported return types. All other data types are returned as a `pandas.DataFrame`.

Function signature[source]

st.data_editor(data, *, width=None, height=None, use_container_width=False, hide_index=None, column_order=None, column_config=None, num_rows="fixed", disabled=False, key=None, on_change=None, args=None, kwargs=None)

column_order (Iterable of str or None)	Specifies the display order of columns. This also affects which columns are visible. For example, `column_order=("col2", "col1")` will display 'col2' first, followed by 'col1', and will hide all other non-index columns. If None (default), the order is inherited from the original data structure.
column_config (dict or None)	Configures how columns are displayed, e.g. their title, visibility, type, or format, as well as editing properties such as min/max value or step. This needs to be a dictionary where each key is a column name and the value is one of: • `None` to hide the column. • A string to set the display label of the column. • One of the column types defined under `st.column_config`, e.g. `st.column_config.NumberColumn("Dollar values", format="$ %d")` to show a column as dollar amounts. See more info on the available column types and config options here. To configure the index column(s), use `_index` as the column name.
num_rows ("fixed" or "dynamic")	Specifies if the user can add and delete rows in the data editor. If "fixed", the user cannot add or delete rows. If "dynamic", the user can add and delete rows in the data editor, but column sorting is disabled. Defaults to "fixed".
disabled (bool or Iterable of str)	Controls the editing of columns. If True, editing is disabled for all columns. If an Iterable of column names is provided (e.g., `disabled=("col1", "col2")`), only the specified columns will be disabled for editing. If False (default), all columns that support editing are editable.
key (str)	An optional string to use as the unique key for this widget. If this is omitted, a key will be generated for the widget based on its content. No two widgets may have the same key.
on_change (callable)	An optional callback invoked when this data_editor's value changes.
args (tuple)	An optional tuple of args to pass to the callback.
kwargs (dict)	An optional dict of kwargs to pass to the callback.

Returns

(pandas.DataFrame, pandas.Series, pyarrow.Table, numpy.ndarray, list, set, tuple, or dict.)	The edited data. The edited data is returned in its original data type if it corresponds to any of the supported return types. All other data types are returned as a `pandas.DataFrame`.

Examples

```
import streamlit as st
import pandas as pd

df = pd.DataFrame(
    [
       {"command": "st.selectbox", "rating": 4, "is_widget": True},
       {"command": "st.balloons", "rating": 5, "is_widget": False},
       {"command": "st.time_input", "rating": 3, "is_widget": True},
   ]
)
edited_df = st.data_editor(df)

favorite_command = edited_df.loc[edited_df["rating"].idxmax()]["command"]
st.markdown(f"Your favorite command is **{favorite_command}** 🎈 ")
```

Built with Streamlit 🎈
Fullscreen *open_in_new*

You can also allow the user to add and delete rows by setting `num_rows` to "dynamic":

```
import streamlit as st
import pandas as pd

df = pd.DataFrame(
    [
       {"command": "st.selectbox", "rating": 4, "is_widget": True},
       {"command": "st.balloons", "rating": 5, "is_widget": False},
       {"command": "st.time_input", "rating": 3, "is_widget": True},
   ]
)
edited_df = st.data_editor(df, num_rows="dynamic")

favorite_command = edited_df.loc[edited_df["rating"].idxmax()]["command"]
st.markdown(f"Your favorite command is **{favorite_command}** 🎈 ")
```

Built with Streamlit 🎈
Fullscreen *open_in_new*

Or you can customize the data editor via `column_config`, `hide_index`, `column_order`, or `disabled`:

```
import pandas as pd
import streamlit as st

df = pd.DataFrame(
    [
        {"command": "st.selectbox", "rating": 4, "is_widget": True},
        {"command": "st.balloons", "rating": 5, "is_widget": False},
        {"command": "st.time_input", "rating": 3, "is_widget": True},
    ]
)
edited_df = st.data_editor(
    df,
    column_config={
        "command": "Streamlit Command",
        "rating": st.column_config.NumberColumn(
            "Your rating",
            help="How much do you like this command (1-5)?",
            min_value=1,
            max_value=5,
            step=1,
            format="%d ⭐",
        ),
        "is_widget": "Widget ?",
    },
    disabled=["command", "is_widget"],
    hide_index=True,
)

favorite_command = edited_df.loc[edited_df["rating"].idxmax()]["command"]
st.markdown(f"Your favorite command is **{favorite_command}** 🎈 ")
```

Built with Streamlit 🎈
Fullscreen *open_in_new*

Configuring columns

You can configure the display and editing behavior of columns in `st.dataframe` and `st.data_editor` via the Column configuration API. We have developed the API to let you add images, charts, and clickable URLs in dataframe and data editor columns. Additionally, you can make individual columns editable, set columns as categorical and specify which options they can take, hide the index of the dataframe, and much more.

Built with Streamlit 🎈
Fullscreen *open_in_new*
←Previous: st.dataframeNext: st.column_config→
forum

Still have questions?

Our forums are full of helpful information and Streamlit experts.

---

forum Ask AI