



Tkinter Treeview

If this Python Tutorial saves you hours of work, please **whitelist it in your ad blocker** 🙏 and

Donate Now

(<https://www.pythontutorial.net/donation/>)

to help us ❤️ pay for the web hosting fee and CDN to keep the

website running.

Summary: in this tutorial, you'll learn about the Tkinter Treeview widget and how to use it to display both tabular and hierarchical data.

Introduction to the Tkinter Treeview widget

A Treeview widget allows you to display data in both tabular and hierarchical structures. To create a Treeview widget, you use the `ttk.Treeview` class:

```
tree = ttk.Treeview(container, **options)
```

A Treeview widget holds a list of items. Each item has one or more columns.

The first column may contain text and an icon that indicates whether it can be expandable or not. The remaining columns contain values of each row.

The first row of the Treeview consists of headings that identify each column by a name.

Using Tkinter Treeview to display tabular data

The following program shows how to use the Treeview widget to display tabular data:

```
import tkinter as tk
from tkinter import ttk
from tkinter.messagebox import showinfo

root = tk.Tk()
root.title('Treeview demo')
root.geometry('620x200')

# define columns
columns = ('first_name', 'last_name', 'email')

tree = ttk.Treeview(root, columns=columns, show='headings')

# define headings
tree.heading('first_name', text='First Name')
tree.heading('last_name', text='Last Name')
tree.heading('email', text='Email')

# generate sample data
contacts = []
for n in range(1, 100):
    contacts.append((f'first {n}', f'last {n}', f'email{n}@example.com'))

# add data to the treeview
for contact in contacts:
    tree.insert('', tk.END, values=contact)

def item_selected(event):
    for selected_item in tree.selection():
        item = tree.item(selected_item)
        record = item['values']
        # show a message
        showinfo(title='Information', message=', '.join(record))
```

```

tree.bind('<<TreeviewSelect>>', item_selected)

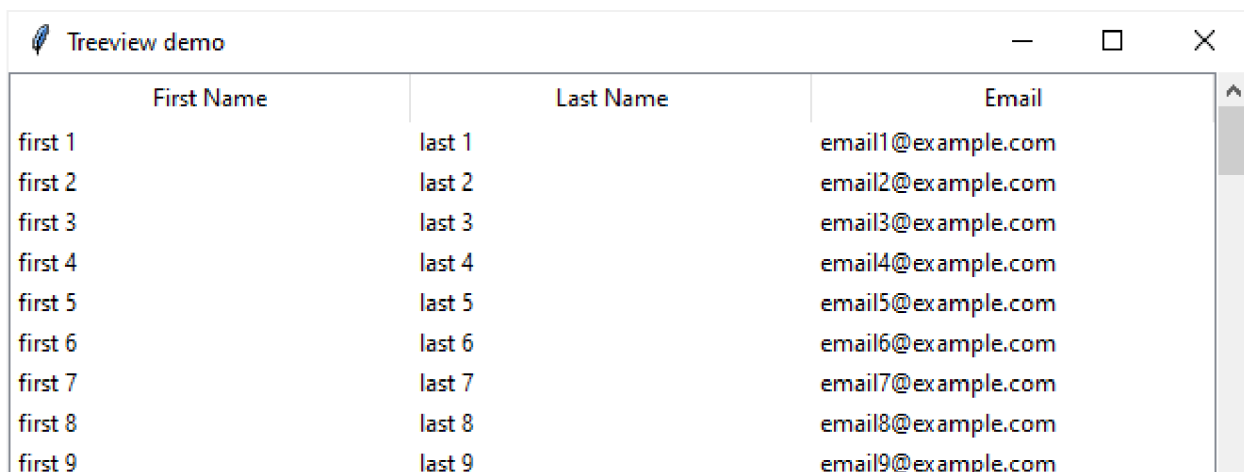
tree.grid(row=0, column=0, sticky='nsew')

# add a scrollbar
scrollbar = ttk.Scrollbar(root, orient=tk.VERTICAL, command=tree.yview)
tree.configure(yscroll=scrollbar.set)
scrollbar.grid(row=0, column=1, sticky='ns')

# run the app
root.mainloop()

```

Output:



| First Name | Last Name | Email |
|------------|-----------|--------------------|
| first 1 | last 1 | email1@example.com |
| first 2 | last 2 | email2@example.com |
| first 3 | last 3 | email3@example.com |
| first 4 | last 4 | email4@example.com |
| first 5 | last 5 | email5@example.com |
| first 6 | last 6 | email6@example.com |
| first 7 | last 7 | email7@example.com |
| first 8 | last 8 | email8@example.com |
| first 9 | last 9 | email9@example.com |

How it works.

First, import `tkinter` module, `ttk` submodule, and the `showinfo` from `tkinter.messagebox` :

```

import tkinter as tk
from tkinter import ttk
from tkinter.messagebox import showinfo

```

Second, create the root window, set its title and size:

```

root = tk.Tk()
root.title('Treeview demo')

```

```
root.geometry('620x200')
```

Third, define identifiers for columns:

```
columns = ('first_name', 'last_name', 'email')
```

Fourth, create a Tkinter's Treeview widget:

```
tree = ttk.Treeview(root, columns=columns, show='headings')
```

In this code, we passed the columns to the `columns` option. The `show='heading'` hides the first column (column `#0`) of the Treeview.

The `show` option accepts one of the following values:

- `'tree'` – shows the column `#0`.
- `'heading'` – shows the header row.
- `'tree headings'` – shows both column `#0` and the header row. This is the default value.
- `''` – doesn't show the column `#0` or the header row.

Fifth, specify the headings for the columns:

```
tree.heading('first_name', text='First Name')
tree.heading('last_name', text='Last Name')
tree.heading('email', text='Email')
```

Sixth, generate a list of tuples for displaying on the Treeview:

```
contacts = []
for n in range(1, 100):
    contacts.append((f'first {n}', f'last {n}', f'email{n}@example.com'))
```

Seventh, create new items, one by one, by using the `insert()` method of the Treeview widget:

```
for contact in contacts:
    tree.insert('', tk.END, values=contact)
```

Eight, define a function to handle the `<>` event. When you select one or more items, the program will show a message box:

```
def item_selected(event):
    for selected_item in tree.selection():
        item = tree.item(selected_item)
        record = item['values']
        # show a message
        showinfo(title='Information', message=', '.join(record))

tree.bind('<<TreeviewSelect>>', item_selected)
```

Ninth, place the Treeview widget on the root window:

```
tree.grid(row=0, column=0, sticky='nsew')
```

Tenth, [add a vertical scrollbar](https://www.pythontutorial.net/tkinter/tkinter-scrollbar/) (<https://www.pythontutorial.net/tkinter/tkinter-scrollbar/>) to the Treeview widget:

```
# add a scrollbar
scrollbar = ttk.Scrollbar(root, orient=tk.VERTICAL, command=tree.yview)
tree.configure(yscroll=scrollbar.set)
scrollbar.grid(row=0, column=1, sticky='ns')
```

Finally, display the root window:

```
root.mainloop()
```

The following program also uses the Treeview widget in the [object-oriented programming](https://www.pythontutorial.net/python-oop/) (<https://www.pythontutorial.net/python-oop/>) approach:

```
import tkinter as tk
from tkinter import ttk
from tkinter.messagebox import showinfo

class App(tk.Tk):
    def __init__(self):
        super().__init__()

        self.title('Treeview demo')
        self.geometry('620x200')

        self.tree = self.create_tree_widget()

    def create_tree_widget(self):
        columns = ('first_name', 'last_name', 'email')
        tree = ttk.Treeview(self, columns=columns, show='headings')

        # define headings
        tree.heading('first_name', text='First Name')
        tree.heading('last_name', text='Last Name')
        tree.heading('email', text='Email')

        tree.bind('<<TreeviewSelect>>', self.item_selected)
        tree.grid(row=0, column=0, sticky=tk.NSEW)

        # add a scrollbar
        scrollbar = ttk.Scrollbar(self, orient=tk.VERTICAL, command=tree.yview)
        tree.configure(yscroll=scrollbar.set)
        scrollbar.grid(row=0, column=1, sticky='ns')

        # generate sample data
        contacts = []
        for n in range(1, 100):
            contacts.append((f'first {n}', f'last {n}', f'email{n}@example.com
```

```

    # add data to the treeview
    for contact in contacts:
        tree.insert('', tk.END, values=contact)

    return tree

def item_selected(self, event):
    for selected_item in self.tree.selection():
        item = self.tree.item(selected_item)
        record = item['values']
        # show a message
        showinfo(title='Information', message=', '.join(record))

if __name__ == '__main__':
    app = App()
    app.mainloop()

```

Adding an item to the Treeview widget

To add an item (or a row) to a Treeview widget, you use the `insert()` method of the `Treeview` widget object. The following example adds an item at the end of the item list:

```
tree.insert('', tk.END, values=contact)
```

To add an item at the beginning of the list, you use zero (`0`) instead of `tk.END` constant:

```
tree.insert('', 0, values=contact)
```

The following program illustrates how to add items to the Treeview:

```

import tkinter as tk
from tkinter import ttk

```

```
class App(tk.Tk):
    def __init__(self):
        super().__init__()

        self.title('Treeview demo')
        self.geometry('620x200')

        self.tree = self.create_tree_widget()

    def create_tree_widget(self):
        columns = ('first_name', 'last_name', 'email')
        tree = ttk.Treeview(self, columns=columns, show='headings')

        # define headings
        tree.heading('first_name', text='First Name')
        tree.heading('last_name', text='Last Name')
        tree.heading('email', text='Email')

        tree.grid(row=0, column=0, sticky=tk.NSEW)

        # adding an item
        tree.insert('', tk.END, values=('John', 'Doe', 'john.doe@email.com'))

        # insert at the end
        tree.insert('', tk.END, values=('Jane', 'Miller', 'jane.miller@email.c

        # insert at the beginning
        tree.insert('', 0, values=('Alice', 'Garcia', 'alice.garcia@email.com')

        return tree

if __name__ == '__main__':
```



```
app = App()
app.mainloop()
```

Deleting items from a Treeview

To delete an item from Treeview, you use the `delete()` method of the Treeview object. The following program shows a Treeview with some items. Clicking an item will delete it from the tree:

```
import tkinter as tk
from tkinter import ttk

class App(tk.Tk):
    def __init__(self):
        super().__init__()

        self.title('Treeview demo')
        self.geometry('620x200')

        self.tree = self.create_tree_widget()

    def create_tree_widget(self):
        columns = ('first_name', 'last_name', 'email')
        tree = ttk.Treeview(self, columns=columns, show='headings')

        # define headings
        tree.heading('first_name', text='First Name')
        tree.heading('last_name', text='Last Name')
        tree.heading('email', text='Email')

        tree.grid(row=0, column=0, sticky=tk.NSEW)

        # adding an item
        tree.insert('', tk.END, values=('John', 'Doe', 'john.doe@email.com'))
```

```

# insert a the end
tree.insert('', tk.END, values=('Jane', 'Miller', 'jane.miller@email.c

# insert at the beginning
tree.insert('', 0, values=('Alice', 'Garcia', 'alice.garcia@email.com'

tree.bind('<<TreeviewSelect>>', self.item_selected)

return tree

def item_selected(self, event):
    for selected_item in self.tree.selection():
        self.tree.delete(selected_item)

if __name__ == '__main__':
    app = App()
    app.mainloop()

```

In this program.

First, bind the item selected event:

```
tree.bind('<<TreeviewSelect>>', self.item_selected)
```

Second, delete the selected item from the tree. To get the selected item, you use the `selection()` method of the `Treeview` object:

```

def item_selected(self, event):
    for selected_item in self.tree.selection():
        self.tree.delete(selected_item)

```

Customizing columns

To change the size of a column and anchor of the item, you can use the `column()` method of the Treeview object:

```
tree.column(size, width, anchor)
```

The following example sets the width for the first name and last name column to 100 and the email to 200. It also set the anchor for the item in each column accordingly:

```
tree.column('first_name', width=100, anchor=tk.W)
tree.column('last_name', width=100, anchor=tk.W)
tree.column('email', width=200, anchor=tk.CENTER)
```

Using Tkinter Treeview to display hierarchical data

The following program illustrates how to use the TreeView widget to display hierarchical data:

```
import tkinter as tk
from tkinter import ttk
from tkinter.messagebox import showinfo

# create root window
root = tk.Tk()
root.title('Treeview Demo - Hierarchical Data')
root.geometry('400x200')

# configure the grid layout
root.rowconfigure(0, weight=1)
root.columnconfigure(0, weight=1)

# create a treeview
tree = ttk.Treeview(root)
tree.heading('#0', text='Departments', anchor=tk.W)
```

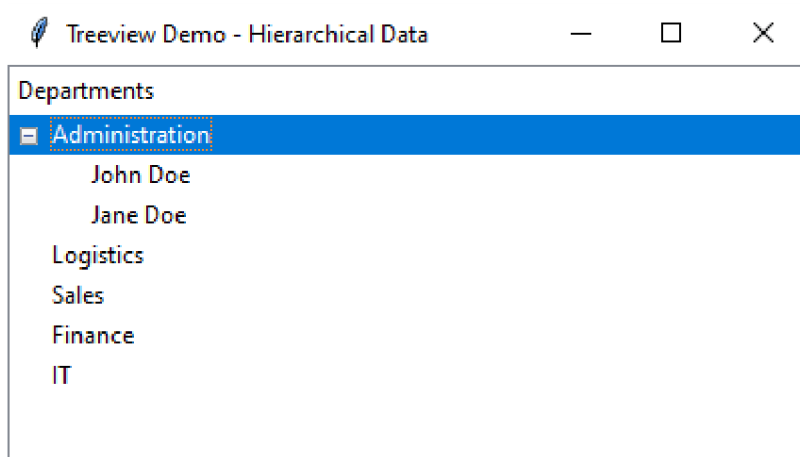
```
# adding data
tree.insert('', tk.END, text='Administration', iid=0, open=False)
tree.insert('', tk.END, text='Logistics', iid=1, open=False)
tree.insert('', tk.END, text='Sales', iid=2, open=False)
tree.insert('', tk.END, text='Finance', iid=3, open=False)
tree.insert('', tk.END, text='IT', iid=4, open=False)

# adding children of first node
tree.insert('', tk.END, text='John Doe', iid=5, open=False)
tree.insert('', tk.END, text='Jane Doe', iid=6, open=False)
tree.move(5, 0, 0)
tree.move(6, 0, 1)

# place the Treeview widget on the root window
tree.grid(row=0, column=0, sticky=tk.NSEW)

# run the app
root.mainloop()
```

Output:



How it works.

We'll focus on the Treeview widget part.

First, create a Treeview widget and set its heading.

```
tree = ttk.Treeview(root)
tree.heading('#0', text='Departments', anchor=tk.W)
```

This Treeview widget has only one column.

Second, add items to the TreeView widget:

```
tree.insert('', tk.END, text='Administration', iid=0, open=False)
tree.insert('', tk.END, text='Logistics', iid=1, open=False)
tree.insert('', tk.END, text='Sales', iid=2, open=False)
tree.insert('', tk.END, text='Finance', iid=3, open=False)
tree.insert('', tk.END, text='IT', iid=4, open=False)
```

Each item is identified by an `iid` . If you skip the `iid` , the insert method will generate one automatically. In this case, you need to have explicit `iid` for adding child items.

Third, add two child items to the item with iid 0 by using the insert() and move() methods:

```
# adding children of first node
tree.insert('', tk.END, text='John Doe', iid=5, open=False)
tree.insert('', tk.END, text='Jane Doe', iid=6, open=False)
tree.move(5, 0, 0)
tree.move(6, 0, 1)
```

Finally, place the Treeview widget on the root window and display it.

```
# place the Treeview widget on the root window
tree.grid(row=0, column=0, sticky=tk.NSEW)

# run the app
root.mainloop()
```

Summary

- Use a Tkinter Treeview widget to display both tabular and hierarchical data.