Listbox リファレンス

複数の項目を一覧表示し、選択を行う `Listbox` ウィジェットについての詳細なリファレンスです。

概要

`Listbox` ウィジェットは、項目の一覧を表示し、ユーザーが一つまたは複数の項目を選択できるコントロールです。スクロール機能も内蔵しており、多数の項目を効率的に表示できます。単一選択、複数選択、範囲選択などの選択モードをサポートします。

基本的な使用方法

シンプルなリストボックス

```
 import tkinter as tk

def on_selection(event):
    selection = listbox.curselection()
    if selection:
        index = selection[0]
        value = listbox.get(index)
        print(f"選択された項目: {value} (インデックス: {index})")

app = tk.Tk()
app.title("Listboxの例")
app.geometry("300x250")

# リストボックスの作成
listbox = tk.Listbox(app, height=6)
listbox.pack(pady=20)

# 項目を追加
items = ["りんご", "みかん", "バナナ", "ぶどう", "いちご", "メロン", "スイカ"]
for item in items:
    listbox.insert(tk.END, item)

# 選択イベントをバインド
listbox.bind("<<ListboxSelect>>", on_selection)

app.mainloop()
```

クラスベースでのリストボックス

```
 import tkinter as tk

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

        self.title("Listboxの例（クラスベース）")
        self.geometry("300x250")

        self.create_widgets()

    def create_widgets(self):
        # リストボックスの作成
        self.listbox = tk.Listbox(self, height=6)
        self.listbox.pack(pady=20)

        # 項目を追加
        items = ["りんご", "みかん", "バナナ", "ぶどう", "いちご", "メロン", "スイカ"]
        for item in items:
            self.listbox.insert(tk.END, item)

        # 選択イベントをバインド
        self.listbox.bind("<<ListboxSelect>>", self.on_selection)

    def on_selection(self, event):
        selection = self.listbox.curselection()
        if selection:
            index = selection[0]
            value = self.listbox.get(index)
            print(f"選択された項目: {value} (インデックス: {index})")

if __name__ == "__main__":
    app = ListboxApp()
    app.mainloop()
```

主要なオプション

オプション	説明
`height`	表示される行数。
`width`	幅を文字数で指定。
`selectmode`	選択モード（`SINGLE`, `BROWSE`, `MULTIPLE`, `EXTENDED`）。
`font`	フォントを指定。
`fg`（または `foreground`）	テキストの色。
`bg`（または `background`）	背景色。
`selectbackground`	選択された項目の背景色。
`selectforeground`	選択された項目のテキスト色。
`relief`	境界線のスタイル（`flat`, `raised`, `sunken`, `groove`, `ridge`）。
`borderwidth`（または `bd`）	境界線の幅。
`activestyle`	アクティブ項目のスタイル（`underline`, `dotbox`, `none`）。
`listvariable`	`tk.StringVar` でリストの内容を管理。

選択モード

モード	説明
`tk.SINGLE`	一度に一つの項目のみ選択可能。
`tk.BROWSE`	一つの項目のみ選択可能だが、ドラッグで選択を変更可能。
`tk.MULTIPLE`	複数の項目を個別に選択/解除可能。
`tk.EXTENDED`	複数選択可能で、Shift/Ctrl キーとの組み合わせで範囲選択可能。

主要なメソッド

メソッド	説明
`insert(index, *elements)`	指定位置に項目を挿入します。
`delete(first, last=None)`	指定範囲の項目を削除します。
`get(first, last=None)`	指定項目の値を取得します。
`curselection()`	現在選択されている項目のインデックスのタプルを返します。
`selection_set(first, last=None)`	指定項目を選択状態にします。
`selection_clear(first, last=None)`	指定項目の選択を解除します。
`size()`	リスト内の項目数を返します。
`see(index)`	指定項目が見えるようにスクロールします。
`activate(index)`	指定項目をアクティブにします。

実用的な例

複数選択とアクション付きリストボックス

```
 import tkinter as tk
from tkinter import messagebox

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

        self.title("複数選択リストボックス")
        self.geometry("400x500")

        self.create_widgets()

    def create_widgets(self):
        # タイトル
        tk.Label(self, text="プログラミング言語を選択（複数可）:", font=("Arial", 12, "bold")).pack(pady=10)

        # リストボックス（複数選択モード）
        self.listbox = tk.Listbox(self, selectmode=tk.MULTIPLE, height=8, font=("Arial", 10))
        self.listbox.pack(pady=10)

        # 項目を追加
        languages = [
            "Python", "JavaScript", "Java", "C++", "C#", "Go", "Rust",
            "TypeScript", "PHP", "Ruby", "Swift", "Kotlin", "Dart"
```

```
        ]
        for lang in languages:
            self.listbox.insert(tk.END, lang)

        # ボタンフレーム
        button_frame = tk.Frame(self)
        button_frame.pack(pady=10)

        # 各種ボタン
        tk.Button(button_frame, text="選択項目を表示", command=self.show_selection).pack(side=tk.LEFT, padx=5)
        tk.Button(button_frame, text="すべて選択", command=self.select_all).pack(side=tk.LEFT, padx=5)
        tk.Button(button_frame, text="選択解除", command=self.clear_selection).pack(side=tk.LEFT, padx=5)

        # 項目操作フレーム
        operation_frame = tk.Frame(self)
        operation_frame.pack(pady=10)

        tk.Label(operation_frame, text="新しい言語:", font=("Arial", 10)).pack(side=tk.LEFT, padx=5)
        self.new_lang_entry = tk.Entry(operation_frame, width=15)
        self.new_lang_entry.pack(side=tk.LEFT, padx=5)

        tk.Button(operation_frame, text="追加", command=self.add_language).pack(side=tk.LEFT, padx=5)
        tk.Button(operation_frame, text="削除", command=self.remove_selected).pack(side=tk.LEFT, padx=5)

        # 結果表示エリア
        tk.Label(self, text="選択結果:", font=("Arial", 10, "bold")).pack(pady=(20, 5))
        self.result_text = tk.Text(self, width=50, height=6, font=("Arial", 9))
        self.result_text.pack(pady=5)

    def show_selection(self):
        selection_indices = self.listbox.curselection()
        if selection_indices:
            selected_items = [self.listbox.get(i) for i in selection_indices]
            result = f"選択された言語 ({len(selected_items)}個):\n"
            result += "\n".join(f"- {item}" for item in selected_items)
        else:
            result = "何も選択されていません。"

        self.result_text.delete(1.0, tk.END)
        self.result_text.insert(1.0, result)

    def select_all(self):
        self.listbox.selection_set(0, tk.END)
        self.show_selection()

    def clear_selection(self):
        self.listbox.selection_clear(0, tk.END)
        self.result_text.delete(1.0, tk.END)

    def add_language(self):
        new_lang = self.new_lang_entry.get().strip()
        if new_lang:
            # 重複チェック
            current_items = [self.listbox.get(i) for i in range(self.listbox.size())]
            if new_lang not in current_items:
                self.listbox.insert(tk.END, new_lang)
                self.new_lang_entry.delete(0, tk.END)
                messagebox.showinfo("追加完了", f"'{new_lang}' を追加しました。")
            else:
                messagebox.showwarning("重複エラー", f"'{new_lang}' は既に存在します。")
        else:
            messagebox.showwarning("入力エラー", "言語名を入力してください。")

    def remove_selected(self):
        selection_indices = list(self.listbox.curselection())
        if selection_indices:
            # 逆順で削除（インデックスのずれを防ぐため）
            for index in reversed(selection_indices):
                self.listbox.delete(index)
            self.result_text.delete(1.0, tk.END)
            messagebox.showinfo("削除完了", f"{len(selection_indices)}個の項目を削除しました。")
        else:
            messagebox.showwarning("選択エラー", "削除する項目を選択してください。")

if __name__ == "__main__":
    app = MultiSelectListboxApp()
    app.mainloop()
```

スクロールバー付きリストボックス

```
import tkinter as tk

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

        self.title("スクロール可能リストボックス")
        self.geometry("400x300")

        self.create_widgets()

    def create_widgets(self):
        # フレームを作成
        list_frame = tk.Frame(self)
        list_frame.pack(fill=tk.BOTH, expand=True, padx=20, pady=20)

        # リストボックスとスクロールバー
```

```
        self.listbox = tk.Listbox(list_frame, selectmode=tk.EXTENDED)
        scrollbar = tk.Scrollbar(list_frame, orient=tk.VERTICAL)

        # スクロールバーとリストボックスを接続
        self.listbox.config(yscrollcommand=scrollbar.set)
        scrollbar.config(command=self.listbox.yview)

        # 配置
        self.listbox.pack(side=tk.LEFT, fill=tk.BOTH, expand=True)
        scrollbar.pack(side=tk.RIGHT, fill=tk.Y)

        # 大量の項目を追加
        for i in range(1, 101):
            self.listbox.insert(tk.END, f"項目 {i:03d}: サンプルデータ {i}")

        # 操作ボタン
        button_frame = tk.Frame(self)
        button_frame.pack(pady=10)

        tk.Button(button_frame, text="先頭へ", command=self.go_to_top).pack(side=tk.LEFT, padx=5)
        tk.Button(button_frame, text="末尾へ", command=self.go_to_bottom).pack(side=tk.LEFT, padx=5)
        tk.Button(button_frame, text="ランダム選択", command=self.random_select).pack(side=tk.LEFT, padx=5)

        # 選択イベント
        self.listbox.bind("<<ListboxSelect>>", self.on_selection)

        # 結果表示
        self.result_label = tk.Label(self, text="項目を選択してください", font=("Arial", 10))
        self.result_label.pack(pady=5)

    def on_selection(self, event):
        selection = self.listbox.curselection()
        if selection:
            if len(selection) == 1:
                index = selection[0]
                value = self.listbox.get(index)
                self.result_label.config(text=f"選択: {value}")
            else:
                self.result_label.config(text=f"{len(selection)}個の項目が選択されています")

    def go_to_top(self):
        self.listbox.see(0)
        self.listbox.selection_clear(0, tk.END)
        self.listbox.selection_set(0)
        self.listbox.activate(0)

    def go_to_bottom(self):
        last_index = self.listbox.size() - 1
        self.listbox.see(last_index)
        self.listbox.selection_clear(0, tk.END)
        self.listbox.selection_set(last_index)
        self.listbox.activate(last_index)

    def random_select(self):
        import random
        size = self.listbox.size()
        if size > 0:
            random_index = random.randint(0, size - 1)
            self.listbox.see(random_index)
            self.listbox.selection_clear(0, tk.END)
            self.listbox.selection_set(random_index)
            self.listbox.activate(random_index)

if __name__ == "__main__":
    app = ScrollableListboxApp()
    app.mainloop()
```

ベストプラクティス

プラクティス	説明
適切な選択モード	用途に応じて適切な `selectmode` を選択します。複数選択が必要な場合は `MULTIPLE` や `EXTENDED` を使用します。
スクロールバーの追加	多数の項目を扱う場合は、スクロールバーを追加してユーザビリティを向上させます。
選択状態の管理	`curselection()` を使用して現在の選択状態を適切に管理します。
項目の動的管理	`insert()` と `delete()` メソッドを使用して項目を動的に追加・削除します。
イベントハンドリング	`<<ListboxSelect>>` イベントを使用して選択変更を検知します。

参考リンク

- Python Docs - tkinter.Listbox
- TkDocs - Listbox