

# Entry リファレンス

一行のテキスト入力を受け付ける `Entry` ウィジェットについての詳細なリファレンスです。

## 概要

`Entry` ウィジェットは、ユーザーが一行のテキストを入力・編集するための基本的なコントロールです。文字列の入力、数値の入力、パスワードの入力などに使用されます。

## 基本的な使用方法

### シンプルなテキスト入力

```
import tkinter as tk

def get_text():
    text = entry.get()
    print(f"入力されたテキスト: {text}")

app = tk.Tk()
app.title("Entryの例")
app.geometry("400x200")

entry = tk.Entry(app, width=30)
entry.pack(pady=20)

button = tk.Button(app, text="テキストを取得", command=get_text)
button.pack()

app.mainloop()
```

### クラスベースでのテキスト入力

```
import tkinter as tk

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

        self.title("Entryの例 (クラスベース) ")
        self.geometry("400x200")

        self.create_widgets()

    def create_widgets(self):
        self.entry = tk.Entry(self, width=30)
        self.entry.pack(pady=20)

        self.button = tk.Button(self, text="テキストを取得", command=self.get_text)
        self.button.pack()

    def get_text(self):
        text = self.entry.get()
        print(f"入力されたテキスト: {text}")

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

## 主要なオプション

| オプション   | 説明   |
|---|--|
| <code>textvariable</code>                       | <code>tk.StringVar</code> などの変数を指定し、その変数と <code>Entry</code> の内容を同期します。  |
| <code>width</code>                              | <code>Entry</code> の幅を文字数で指定します。   |
| <code>font</code>                               | フォントを指定。タプル ( <code>"フォント名"</code> , <code>サイズ</code> , <code>"スタイル"</code> ) や文字列で指定。                                 |
| <code>fg</code> (または <code>foreground</code> )  | テキストの色。  |
| <code>bg</code> (または <code>background</code> )  | <code>Entry</code> の背景色。   |
| <code>relief</code>                             | 境界線のスタイル ( <code>flat</code> , <code>raised</code> , <code>sunken</code> , <code>groove</code> , <code>ridge</code> )。 |
| <code>borderwidth</code> (または <code>bd</code> ) | 境界線の幅。   |
| <code>state</code>                              | <code>Entry</code> の状態 ( <code>normal</code> , <code>readonly</code> , <code>disabled</code> )。                        |
| <code>show</code>                               | 入力文字を隠す文字を指定 (パスワード入力など)。例: <code>show="*"</code>  |

| オプション                        | 説明   |
|------------------------------|--|
| <code>justify</code>         | テキストの配置 ( <code>left</code> , <code>center</code> , <code>right</code> )。  |
| <code>validate</code>        | 入力値の検証タイミング ( <code>none</code> , <code>focus</code> , <code>focusin</code> , <code>focusout</code> , <code>key</code> , <code>all</code> )。 |
| <code>validatecommand</code> | 検証時に実行される関数。   |

## 主要なメソッド

| メソッド                                  | 説明   |
|---------------------------------------|--|
| <code>get()</code>                    | Entry の現在のテキストを取得します。                              |
| <code>set(value)</code>               | Entry のテキストを設定します。( <code>textvariable</code> 使用時) |
| <code>insert(index, string)</code>    | 指定位置にテキストを挿入します。                                   |
| <code>delete(first, last=None)</code> | 指定範囲のテキストを削除します。                                   |
| <code>select_range(start, end)</code> | 指定範囲のテキストを選択します。                                   |
| <code>select_clear()</code>           | テキストの選択を解除します。                                     |
| <code>icursor(index)</code>           | カーソルを指定位置に移動します。                                   |

## 実用的な例

### フォームアプリケーション

```
import tkinter as tk
from tkinter import messagebox

class FormApp(tk.Tk):
    def __init__(self):
        super().__init__()
        self.title("フォーム入力")
        self.geometry("400x300")

        self.create_widgets()

    def create_widgets(self):
        # 名前入力
        tk.Label(self, text="名前:", font=("Arial", 12)).pack(pady=(20, 5))
        self.name_var = tk.StringVar()
        self.name_entry = tk.Entry(self, textvariable=self.name_var, width=30, font=("Arial", 11))
        self.name_entry.pack()

        # 年齢入力
        tk.Label(self, text="年齢:", font=("Arial", 12)).pack(pady=(10, 5))
        self.age_var = tk.StringVar()
        self.age_entry = tk.Entry(self, textvariable=self.age_var, width=30, font=("Arial", 11))
        self.age_entry.pack()

        # パスワード入力
        tk.Label(self, text="パスワード:", font=("Arial", 12)).pack(pady=(10, 5))
        self.password_var = tk.StringVar()
        self.password_entry = tk.Entry(
            self,
            textvariable=self.password_var,
            width=30,
            font=("Arial", 11),
            show="*" # パスワードを隠す
        )
        self.password_entry.pack()

        # ボタン
        button_frame = tk.Frame(self)
        button_frame.pack(pady=20)

        submit_button = tk.Button(button_frame, text="送信", command=self.submit_form)
        submit_button.pack(side=tk.LEFT, padx=5)

        clear_button = tk.Button(button_frame, text="クリア", command=self.clear_form)
        clear_button.pack(side=tk.LEFT, padx=5)

        # 初期フォーカスを名前入力に設定
        self.name_entry.focus()

    def submit_form(self):
        name = self.name_var.get().strip()
        age = self.age_var.get().strip()
        password = self.password_var.get()

        if not name:
            messagebox.showerror("エラー", "名前を入力してください。")
            self.name_entry.focus()
            return

        if not age:
            messagebox.showerror("エラー", "年齢を入力してください。")
            self.age_entry.focus()
            return
```

```
try:
    age_int = int(age)
    if age_int < 0 or age_int > 150:
        raise ValueError
except ValueError:
    messagebox.showerror("エラー", "有効な年齢を入力してください。")
    self.age_entry.focus()
    return

if not password:
    messagebox.showerror("エラー", "パスワードを入力してください。")
    self.password_entry.focus()
    return

messagebox.showinfo("送信完了", f"名前: {name}\n年齢: {age}\nパスワードは設定されました。")

def clear_form(self):
    self.name_var.set("")
    self.age_var.set("")
    self.password_var.set("")
    self.name_entry.focus()

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

## 入力値のリアルタイム検証

```
import tkinter as tk

class ValidatedEntryApp(tk.Tk):
    def __init__(self):
        super().__init__()
        self.title("入力値検証")
        self.geometry("400x300")

        # 検証関数を登録
        self.validate_number = self.register(self.validate_number_input)

        self.create_widgets()

    def validate_number_input(self, value):
        """数値のみを許可する検証関数"""
        if value == "":
            return True # 空文字は許可
        try:
            float(value)
            return True
        except ValueError:
            return False

    def create_widgets(self):
        # 通常の Entry
        tk.Label(self, text="自由入力:", font=("Arial", 12)).pack(pady=(20, 5))
        self.free_entry = tk.Entry(self, width=30, font=("Arial", 11))
        self.free_entry.pack()

        # 数値のみの Entry
        tk.Label(self, text="数値のみ:", font=("Arial", 12)).pack(pady=(10, 5))
        self.number_entry = tk.Entry(
            self,
            width=30,
            font=("Arial", 11),
            validate='key', # キー入力時に検証
            validatecommand=(self.validate_number, '%P') # %P は新しい値
        )
        self.number_entry.pack()

        # 読み取り専用の Entry
        tk.Label(self, text="読み取り専用:", font=("Arial", 12)).pack(pady=(10, 5))
        self.readonly_var = tk.StringVar(value="変更できません")
        self.readonly_entry = tk.Entry(
            self,
            textvariable=self.readonly_var,
            width=30,
            font=("Arial", 11),
            state='readonly',
            bg='lightgray'
        )
        self.readonly_entry.pack()

        # 結果表示
        tk.Label(self, text="結果:", font=("Arial", 12)).pack(pady=(20, 5))
        self.result_text = tk.Text(self, width=40, height=6, font=("Arial", 10))
        self.result_text.pack()

        # 値を取得するボタン
        get_button = tk.Button(self, text="値を取得", command=self.get_values)
        get_button.pack(pady=10)

    def get_values(self):
        free_value = self.free_entry.get()
        number_value = self.number_entry.get()
        readonly_value = self.readonly_var.get()

        result = f"自由入力: {free_value}\n"
        result += f"数値入力: {number_value}\n"
```

```
result += f"読み取り専用: {readonly_value}\n"

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

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

## ベストプラクティス

| プラクティス                  | 説明  |
|-------------------------|---|
| <b>textvariable の活用</b> | <code>tk.StringVar</code> を使用することで、Entry の値とアプリケーションの状態を簡単に同期できます。          |
| <b>入力値の検証</b>           | <code>validate</code> と <code>validatecommand</code> オプションを使用して、不正な入力を防ぎます。 |
| <b>フォーカス管理</b>          | <code>focus()</code> メソッドを使用して、適切なフィールドにフォーカスを設定します。                        |
| <b>エラーハンドリング</b>        | ユーザー入力を処理する際は、適切なエラーハンドリングとユーザーフィードバックを提供します。                               |
| <b>パスワード入力</b>          | 機密情報の入力には <code>show</code> オプションを使用して文字を隠します。                              |

## 参考リンク

- [Python Docs - tkinter.Entry](#)
- [TkDocs - Entry](#)