# QLineEdit クラス

## 概要

`QLineEdit` は、1行のテキスト入力を可能にするウィジェットです。ユーザーがテキストを入力・編集するための基本的なコンポーネントで、フォーム、検索ボックス、設定画面などで広く使用されます。

## 基本的な使用方法

```
 from PySide6.QtWidgets import QApplication, QLineEdit, QVBoxLayout, QWidget
import sys

app = QApplication(sys.argv)
window = QWidget()
layout = QVBoxLayout()

# 基本的なQLineEditの作成
line_edit = QLineEdit()
line_edit.setPlaceholderText("ここにテキストを入力")
layout.addWidget(line_edit)

window.setLayout(layout)
window.show()
sys.exit(app.exec())
```

## 主要なプロパティとメソッド

### テキスト操作

| メソッド | 説明 | 例 |
|---|---|---|
| `setText(text)` | テキストを設定 | `line_edit.setText("Hello")` |
| `text()` | 現在のテキストを取得 | `current_text = line_edit.text()` |
| `clear()` | テキストをクリア | `line_edit.clear()` |
| `insert(text)` | カーソル位置にテキストを挿入 | `line_edit.insert("World")` |

### プレースホルダーテキスト

| メソッド | 説明 | 例 |
|---|---|---|
| `setPlaceholderText(text)` | プレースホルダーテキストを設定 | `line_edit.setPlaceholderText("名前を入力")` |
| `placeholderText()` | プレースホルダーテキストを取得 | `placeholder = line_edit.placeholderText()` |

### 入力制限と検証

| メソッド | 説明 | 例 |
|---|---|---|
| `setMaxLength(length)` | 最大文字数を設定 | `line_edit.setMaxLength(50)` |
| `maxLength()` | 最大文字数を取得 | `max_len = line_edit.maxLength()` |
| `setValidator(validator)` | 入力バリデーターを設定 | `line_edit.setValidator(QIntValidator())` |
| `setInputMask(mask)` | 入力マスクを設定 | `line_edit.setInputMask("000.000.000.000")` |

### 表示モード

| メソッド | 説明 | 例 |
|---|---|---|
| `setEchoMode(mode)` | 文字の表示モードを設定 | `line_edit.setEchoMode(QLineEdit.Password)` |
| `echoMode()` | 現在の表示モードを取得 | `mode = line_edit.echoMode()` |
| `setReadOnly(readonly)` | 読み取り専用モードの設定 | `line_edit.setReadOnly(True)` |
| `isReadOnly()` | 読み取り専用かどうかを確認 | `is_readonly = line_edit.isReadOnly()` |

### カーソルとテキスト選択

| メソッド | 説明 | 例 |
|---|---|---|
| `setCursorPosition(pos)` | カーソル位置を設定 | `line_edit.setCursorPosition(5)` |

| メソッド | 説明 | 例 |
|---|---|---|
| `cursorPosition()` | 現在のカーソル位置を取得 | `pos = line_edit.cursorPosition()` |
| `selectAll()` | すべてのテキストを選択 | `line_edit.selectAll()` |
| `setSelection(start, length)` | 指定範囲のテキストを選択 | `line_edit.setSelection(0, 5)` |
| `selectedText()` | 選択されたテキストを取得 | `selected = line_edit.selectedText()` |

## エコーモード

QLineEditは、以下のエコーモードをサポートします：

| モード | 説明 | 用途 |
|---|---|---|
| `Normal` | 通常表示（デフォルト） | 一般的なテキスト入力 |
| `NoEcho` | 文字を表示しない | 機密情報の入力 |
| `Password` | アスタリスクで表示 | パスワード入力 |
| `PasswordEchoOnEdit` | 編集時のみ文字を表示 | ユーザビリティを考慮したパスワード入力 |

## 主要なシグナル

| シグナル | 説明 | 使用例 |
|---|---|---|
| `textChanged(text)` | テキストが変更された時 | `line_edit.textChanged.connect(on_text_changed)` |
| `textEdited(text)` | ユーザーがテキストを編集した時 | `line_edit.textEdited.connect(on_text_edited)` |
| `returnPressed()` | Enterキーが押された時 | `line_edit.returnPressed.connect(on_enter_pressed)` |
| `editingFinished()` | 編集が完了した時 | `line_edit.editingFinished.connect(on_edit_finished)` |
| `selectionChanged()` | テキスト選択が変更された時 | `line_edit.selectionChanged.connect(on_selection_changed)` |

## 実用的な使用例

### 1. パスワード入力フィールド

```
password_edit = QLineEdit()
password_edit.setEchoMode(QLineEdit.Password)
password_edit.setPlaceholderText("パスワードを入力")
```

### 2. 数値のみの入力

```
from PySide6.QtGui import QIntValidator

number_edit = QLineEdit()
number_edit.setValidator(QIntValidator(0, 100))  # 0-100の整数のみ
number_edit.setPlaceholderText("0-100の数値を入力")
```

### 3. IPアドレス入力

```
ip_edit = QLineEdit()
ip_edit.setInputMask("000.000.000.000;_")
ip_edit.setPlaceholderText("192.168.1.1")
```

### 4. リアルタイム検索

```
search_edit = QLineEdit()
search_edit.setPlaceholderText("検索...")
search_edit.textChanged.connect(perform_search)

def perform_search(text):
    # 検索処理を実行
    print(f"検索中: {text}")
```

## ベストプラクティス

### 1. 適切なプレースホルダーの使用

```
# 良い例
line_edit.setPlaceholderText("例: user@example.com")
```

```
# 避けるべき例
line_edit.setPlaceholderText("テキストを入力")
```

### 2. 入力検証の実装

```
# 入力後の検証
def validate_email(text):
    if "@" not in text:
        line_edit.setStyleSheet("border: 2px solid red")
    else:
        line_edit.setStyleSheet("border: 2px solid green")

line_edit.textChanged.connect(validate_email)
```

### 3. 適切なサイズ設定

```
# 固定幅の設定
line_edit.setFixedWidth(200)

# 最小・最大幅の設定
line_edit.setMinimumWidth(100)
line_edit.setMaximumWidth(300)
```

## 注意事項

1. **パフォーマンス**: `textChanged` シグナルは文字入力のたびに発火するため、重い処理は避ける
2. **バリデーション**: ユーザー体験を考慮して、リアルタイム検証と最終検証を適切に使い分ける
3. **アクセシビリティ**: 視覚的な手がかりだけでなく、プレースホルダーテキストで機能を明確に説明する

## 関連するクラス

- **QTextEdit**: 複数行のテキスト編集
- **QValidator**: 入力検証用のベースクラス
- **QCompleter**: オートコンプリート機能
- **QLabel**: ラベル表示（QLineEditと組み合わせてフォーム作成）

## 参考リンク

- Qt公式ドキュメント - QLineEdit
- PySide6公式ドキュメント