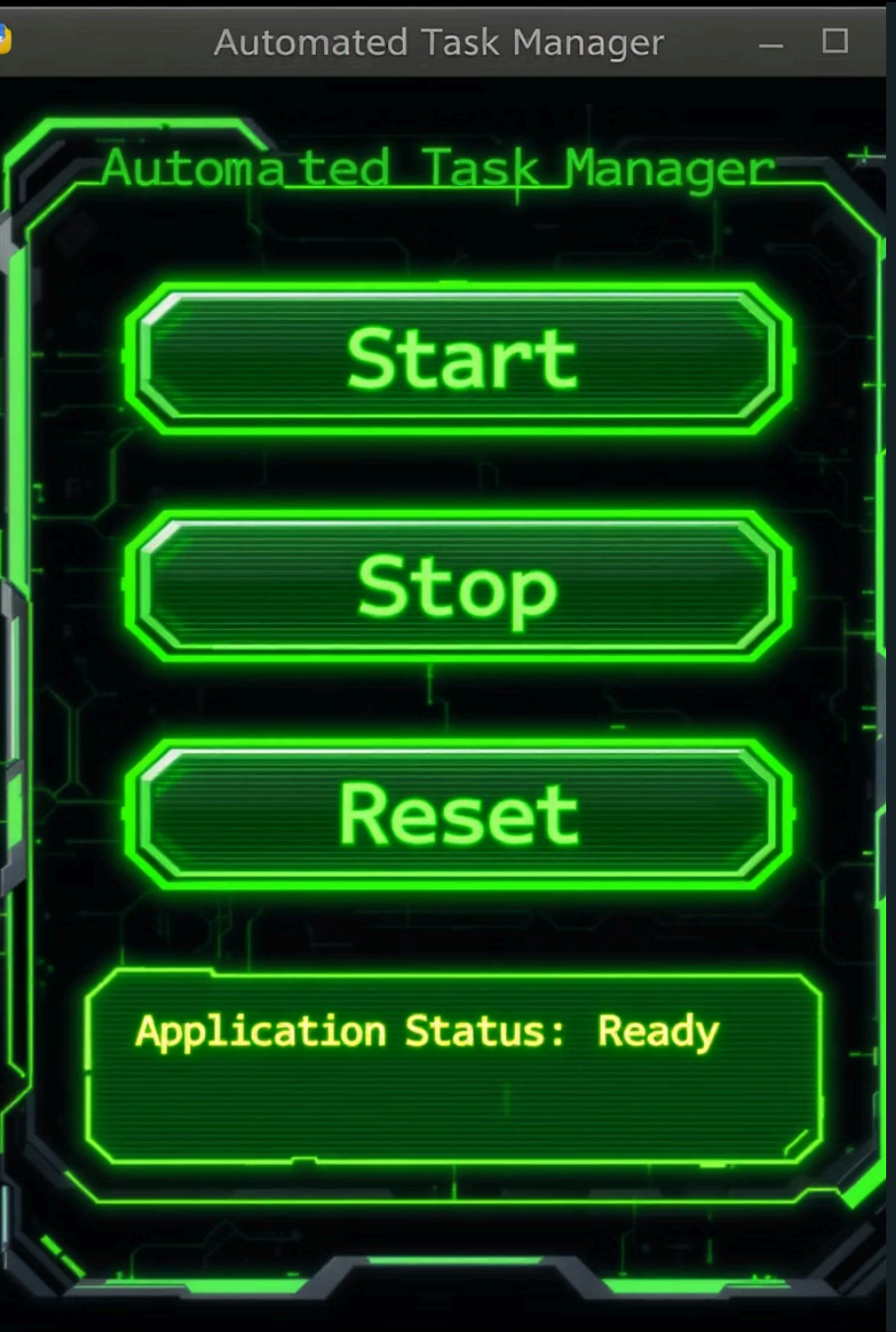


Programmazione di Risposte agli Eventi in Python Tkinter

Benvenuti a questo corso sulla programmazione di eventi in Python Tkinter. In questa presentazione esploreremo come gestire interazioni utente come click, pressioni di tasti e selezioni all'interno delle nostre interfacce grafiche.

La programmazione basata su eventi è il cuore di qualsiasi interfaccia utente moderna. Imparerete come catturare e rispondere a una varietà di azioni dell'utente, rendendo le vostre applicazioni dinamiche e interattive.

Questo corso è ideale per sviluppatori con conoscenze base di Python che desiderano approfondire lo sviluppo di interfacce grafiche reattive e funzionali.

Rilevazione del Click su un Pulsante

Concetti Base

In Tkinter, la gestione dei click sui pulsanti avviene tramite il parametro **command**. Questo parametro accetta una funzione che verrà eseguita quando l'utente clicca sul pulsante.

È possibile definire funzioni separate o utilizzare funzioni lambda per gestire i click con maggiore flessibilità, specialmente quando si devono passare parametri aggiuntivi.

```
1
1  Tkinter button(event event)()
2  {
3  flckcle(mt_list()
4  bister wipen_tost_faper(ifst_label' (hebl (when'cliope'
4  clearfont_text)
5  label tp:(lanbler'lick(): {
8    imt me text cloxt; {
19   ixt fus clicked
11 }
13 ;
14 :
15 :
```

Il collegamento tra l'evento click e la funzione di callback viene stabilito al momento della creazione del pulsante. È importante che la funzione indicata nel parametro command non includa le parentesi, altrimenti verrebbe eseguita immediatamente anziché al click.

Rilevazione del Click in un'Area Vuota

Binding dell'Evento

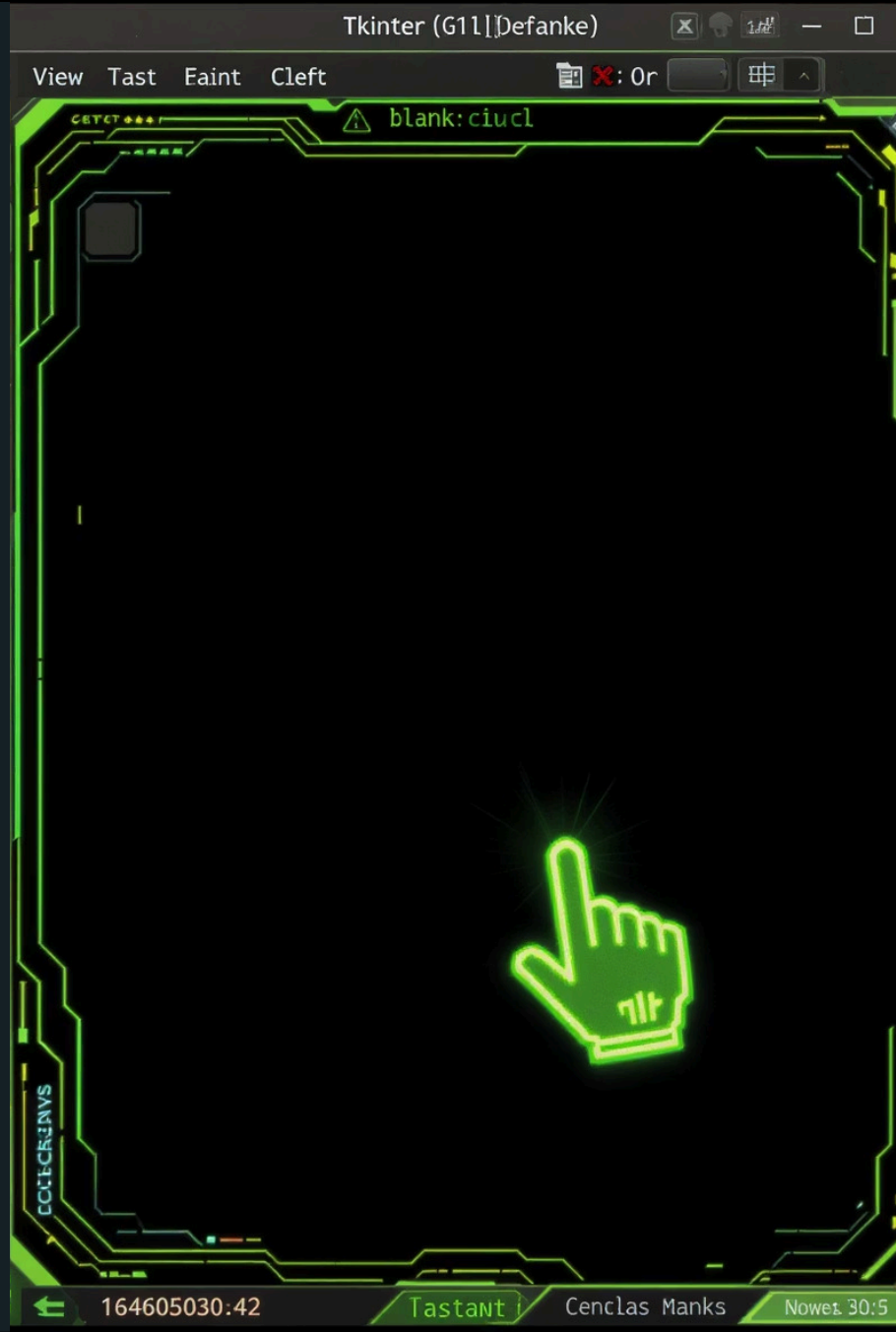
Per rilevare i click in aree vuote o su widget che non hanno il parametro command, utilizziamo il metodo **bind()** per associare l'evento <Button-1> a una funzione.

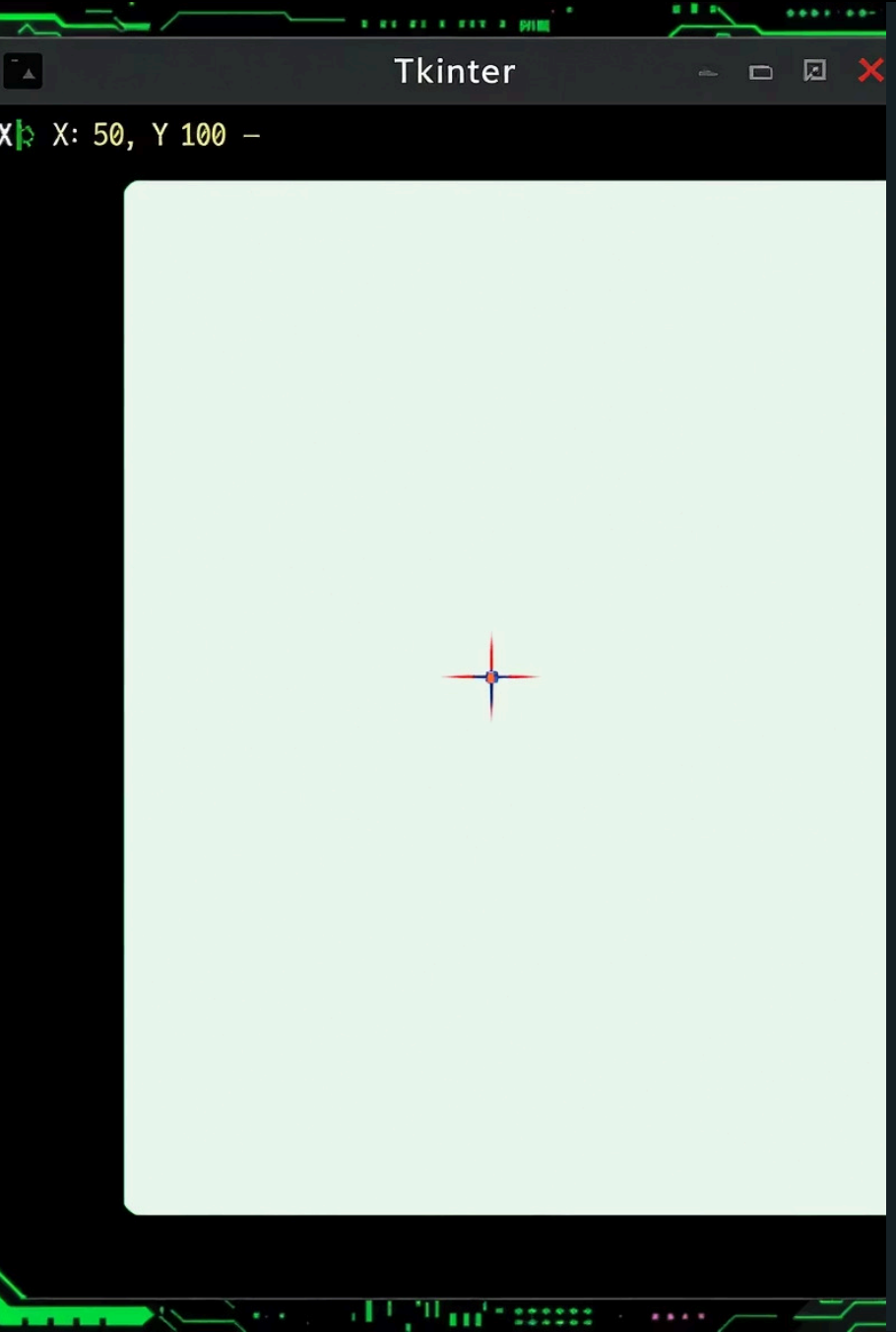
Implementazione

Si può applicare il binding a qualsiasi widget, inclusi Frame, Canvas o Label. L'evento <Button-1> rappresenta il click sinistro, mentre <Button-2> e <Button-3> rappresentano rispettivamente il click centrale e destro.

Funzione di Callback

La funzione che gestisce l'evento riceve automaticamente un parametro (spesso chiamato "event" o "e") che contiene informazioni dettagliate sull'evento stesso.





Visualizzazione delle Coordinate del Click

Attributi dell'Evento

L'oggetto evento passato alla funzione di callback contiene numerose informazioni utili, tra cui le coordinate del punto in cui è avvenuto il click attraverso gli attributi **x** e **y**.

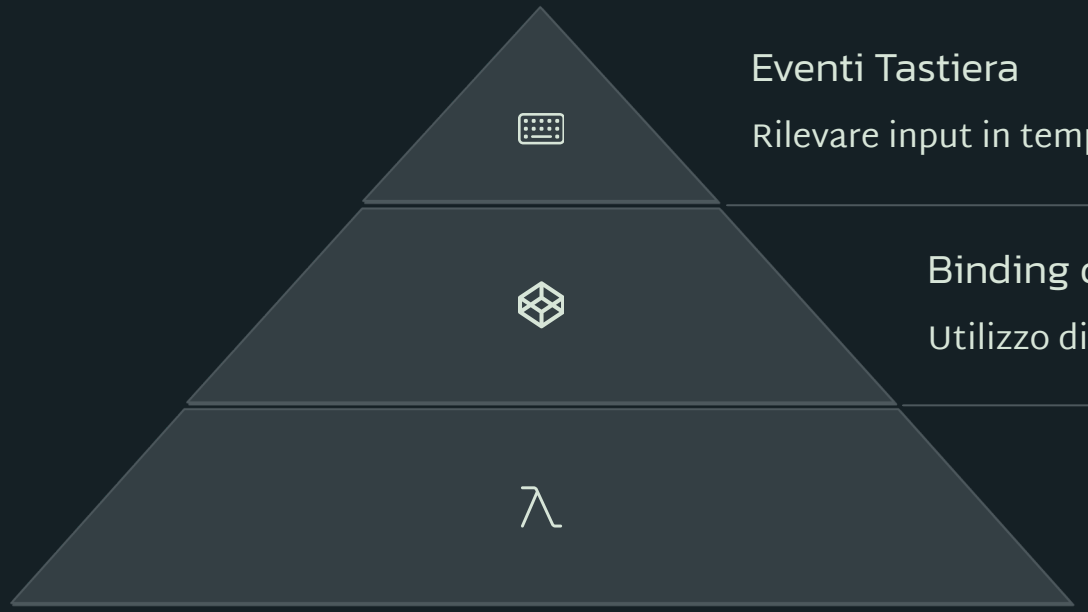
Coordinare Relative e Assolute

Le coordinate x e y sono relative al widget su cui è avvenuto il click. Per ottenere le coordinate assolute rispetto alla finestra, si possono utilizzare **event.x_root** e **event.y_root**.

Applicazioni Pratiche

La rilevazione delle coordinate è fondamentale per applicazioni di disegno, posizionamento di elementi, creazione di mappe interattive e implementazione di funzionalità drag-and-drop.

Rilevazione di Pressione Tasti in Entry



Eventi Tastiera

Rilevare input in tempo reale

Binding dei Tasti

Utilizzo di <KeyPress> e <Key>

Elaborazione Eventi

Analisi di event.char, event.keysym e event.keycode

Nei widget Entry di Tkinter, possiamo intercettare la pressione di tasti usando il metodo bind() con l'evento <KeyPress> o semplicemente <Key>. Questo ci permette di elaborare input in tempo reale, ad esempio per validare i dati mentre l'utente digita, invece di aspettare la conferma.

L'oggetto evento fornisce tre attributi utili: event.char (il carattere digitato), event.keysym (il nome simbolico del tasto) e event.keycode (il codice numerico del tasto). Questi attributi sono fondamentali per implementare filtri e controlli avanzati sull'input.

Aggiunta di Logica su Invio (<Return>)

Binding Specifico

Associare l'evento <Return> a una funzione per intercettare la pressione del tasto Invio

Callback di Validazione

Implementare la logica di validazione e processamento dell'input quando l'utente preme Invio

Aggiornamento UI

Aggiornare l'interfaccia in risposta all'input confermato dall'utente

L'evento <Return> è particolarmente utile nei form di inserimento dati, dove vogliamo elaborare l'input dell'utente quando preme il tasto Invio. Questo approccio rende l'interfaccia più naturale e intuitiva, consentendo agli utenti di confermare rapidamente i dati inseriti senza dover necessariamente cliccare su un pulsante.

Il codice per implementare questa funzionalità è semplice ma potente: entry.bind('<Return>', funzione_di_callback). La funzione di callback può poi eseguire operazioni come la validazione dell'input, l'invio di dati a un server, o l'aggiornamento di altri elementi dell'interfaccia.

Cambio Dinamico di Testo con un Click

Evento Click

Utilizziamo l'evento <Button-1> o il parametro command per rilevare quando l'utente clicca su un elemento dell'interfaccia.

Modifica del Testo

Con i metodi config() o configure(), possiamo modificare dinamicamente il testo di Label, Button e altri widget.

Variabili Tkinter

Le StringVar, IntVar e altre variabili Tkinter consentono un aggiornamento automatico dei widget collegati quando il loro valore cambia.

La capacità di modificare dinamicamente il contenuto testuale dei widget è essenziale per creare interfacce interattive. Ad esempio, un pulsante può cambiare da "Avvia" a "Interrompi" dopo essere stato cliccato, fornendo un feedback visivo immediato all'utente sull'azione eseguita.

L'uso di variabili Tkinter come StringVar è spesso il metodo più elegante per gestire questi cambiamenti, poiché offrono un meccanismo di aggiornamento automatico: basta modificare il valore della variabile e tutti i widget collegati si aggiorneranno di conseguenza.

Evidenziare un Widget al Passaggio del Mouse

Evento <Enter>

Si attiva quando il mouse entra nell'area del widget

Cambio Stile

Modifica dell'aspetto del widget tramite configure()

Ripristino Stile

Ritorno all'aspetto originale del widget

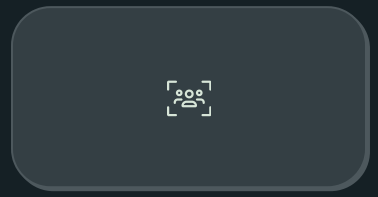
Evento <Leave>

Si attiva quando il mouse esce dall'area del widget

L'effetto hover (evidenziazione al passaggio del mouse) è una tecnica comune per migliorare l'usabilità dell'interfaccia. In Tkinter, questo effetto si realizza combinando gli eventi <Enter> e <Leave> con il metodo configure() per modificare proprietà come background, foreground, o font dei widget.

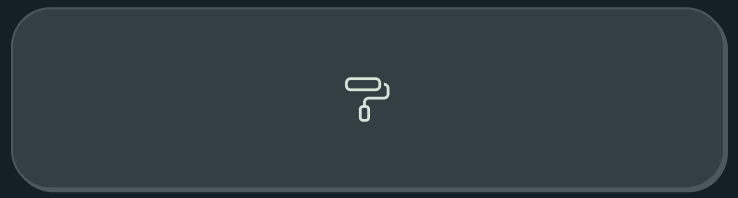
Questa tecnica è particolarmente efficace per pulsanti, link e aree cliccabili, poiché fornisce un feedback visivo che aiuta l'utente a identificare gli elementi interattivi dell'interfaccia. Un'implementazione comune include l'aumento della luminosità o il cambio di colore quando il mouse passa sopra un elemento.

Colorare un Campo al Ricevimento del Focus



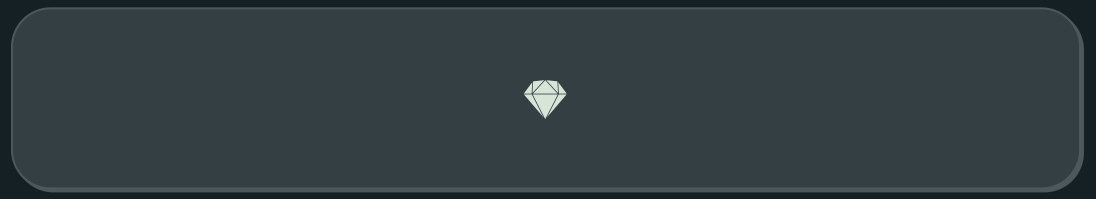
Eventi di Focus

<FocusIn> e <FocusOut>



Modifica Colore

Cambio di background o foreground



Feedback Visivo

Migliore esperienza utente

Il focus è lo stato che indica quale widget sta attualmente ricevendo l'input da tastiera. In Tkinter, possiamo utilizzare gli eventi <FocusIn> e <FocusOut> per rilevare quando un widget riceve o perde il focus, rispettivamente.

Cambiare il colore di un campo quando riceve il focus è una tecnica efficace per guidare l'attenzione dell'utente e indicare chiaramente quale elemento è attualmente attivo. Questo è particolarmente utile nei form con molti campi di input, dove è importante che l'utente sappia esattamente dove sta digitando. L'implementazione richiede semplicemente di collegare funzioni di callback agli eventi di focus che modificano gli attributi di colore del widget.

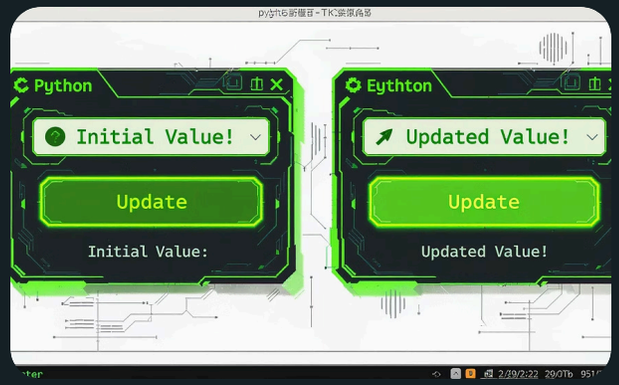
Uso di Lambda per Passare Argomenti

Problema	Le funzioni specificate in command o bind non possono ricevere parametri personalizzati
Soluzione	Utilizzo di funzioni lambda come wrapper
Sintassi Base	command=lambda: funzione(parametro)
Con Evento	widget.bind('<Event>', lambda e: funzione(e, parametro))
Vantaggio	Riutilizzo della stessa funzione con parametri diversi

Le funzioni lambda sono fondamentali nella programmazione di eventi in Tkinter quando è necessario passare parametri aggiuntivi a una funzione di callback. Senza lambda, sarebbe impossibile specificare quali argomenti passare alla funzione quando l'evento si verifica.

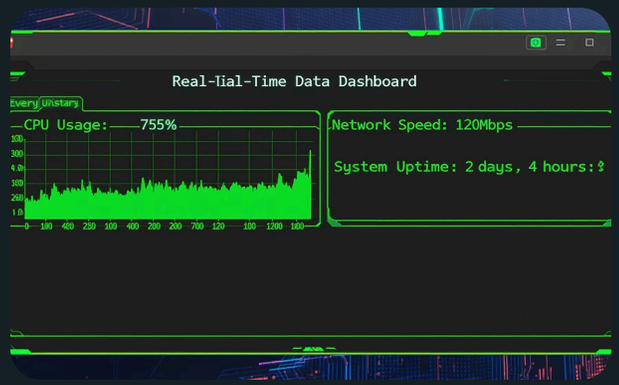
Un caso d'uso comune è quando si hanno più pulsanti che devono chiamare la stessa funzione ma con parametri diversi. Ad esempio, una calcolatrice con pulsanti numerici può utilizzare lambda per passare il valore specifico di ciascun pulsante alla funzione di gestione: button1 = Button(root, text="1", command=lambda: aggiungi_numero(1)).

Aggiornamento di una Label con Evento



Aggiornamento Diretto

Il metodo più semplice per aggiornare una Label è utilizzare il metodo config() o configure() per modificare il testo mostrato. Questo approccio è ideale per aggiornamenti puntuali in risposta a eventi specifici.



Aggiornamento con Variabili

Per un'interfaccia più reattiva, è possibile collegare la Label a una StringVar e aggiornare il valore della variabile. Questo metodo è particolarmente utile quando più widget devono riflettere lo stesso valore.

Messaggi di Stato

Le Label vengono spesso utilizzate per mostrare messaggi di stato, feedback o risultati di operazioni. Un'interfaccia ben progettata dovrebbe aggiornare queste informazioni in tempo reale per mantenere l'utente informato.

Disabilitare un Pulsante Dopo il Click

Click sul Pulsante

L'utente attiva l'evento cliccando sul pulsante

Esecuzione Funzione

Si esegue la funzione associata al comando

Disabilitazione

Il pulsante viene disabilitato con state='disabled'

Feedback Visivo

Il pulsante appare in grigio per indicare lo stato disabilitato

Disabilitare un pulsante dopo il click è una tecnica importante per prevenire azioni duplicate o ripetute che potrebbero causare problemi, come l'invio multiplo di un modulo o l'avvio di più istanze di un processo.

L'implementazione è semplice: nella funzione di callback associata al pulsante, dopo aver eseguito l'operazione principale, si aggiunge una riga di codice che imposta lo stato del pulsante su 'disabled' utilizzando il metodo configure(). Tkinter si occuperà automaticamente di modificare l'aspetto visivo del pulsante per indicare all'utente che non è più cliccabile.

Rilevazione di Doppio Click su Listbox

1

Binding Evento

Collegare l'evento <Double-Button-1> alla Listbox per rilevare i doppi click

2

Indice Selezionato

Utilizzare curselection() per ottenere l'indice dell'elemento selezionato

3

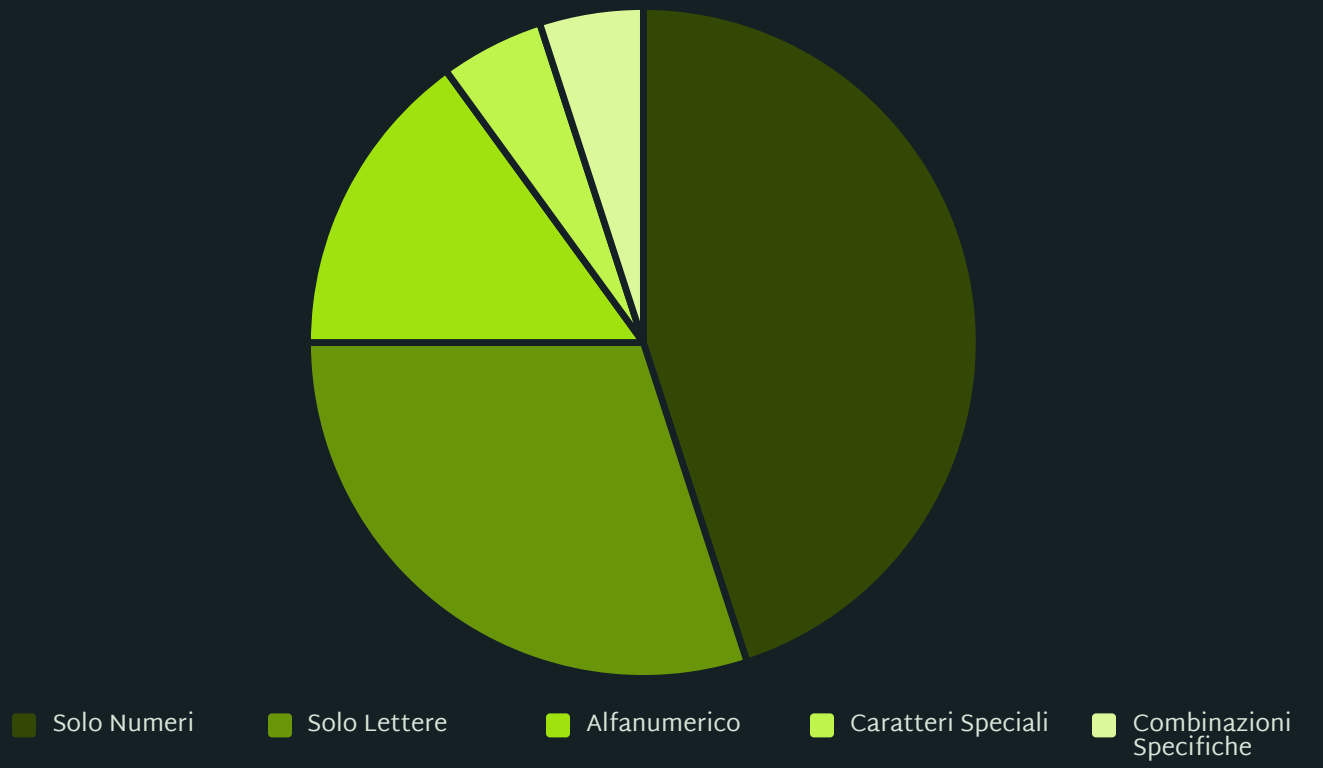
Recupero Valore

Ottenere il valore effettivo dell'elemento con get(indice)

Il doppio click è un'interazione comune utilizzata per attivare un'azione secondaria su un elemento selezionato. Nelle Listbox di Tkinter, questa funzionalità è particolarmente utile per implementare azioni come "apri elemento selezionato" o "modifica elemento selezionato".

Per recuperare l'elemento su cui è avvenuto il doppio click, è necessario combinare il metodo curselection() (che restituisce l'indice dell'elemento selezionato) con il metodo get() (che recupera il valore effettivo dato l'indice). Questo permette di implementare azioni specifiche basate sull'elemento selezionato dall'utente.

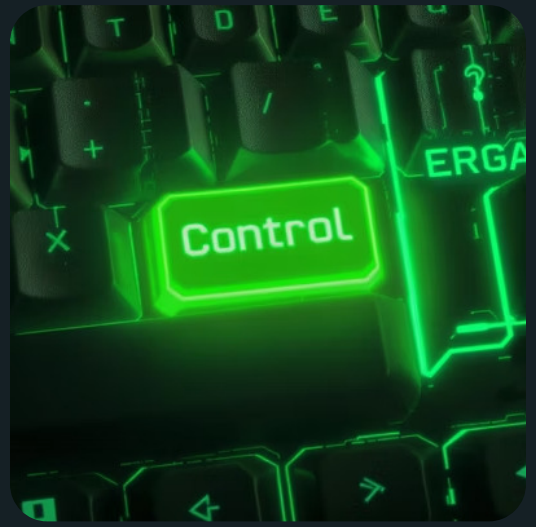
Utilizzo di event.char per Filtrare Tasti

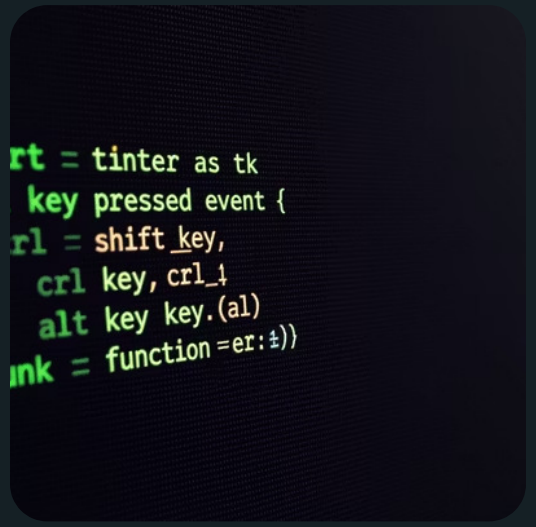


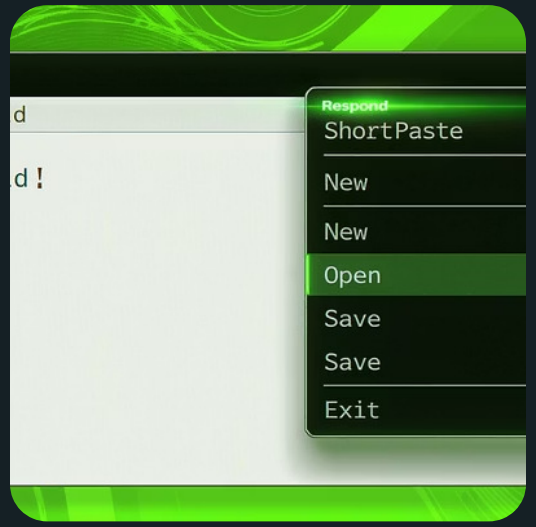
Il filtraggio dell'input da tastiera è una tecnica essenziale per migliorare la validità dei dati inseriti dall'utente. Utilizzando l'attributo event.char, possiamo esaminare ciascun carattere digitato e decidere se accettarlo o bloccarlo prima che venga inserito nel widget.

Un approccio comune è intercettare l'evento <KeyPress> e utilizzare espressioni regolari o semplici controlli condizionali per verificare se il carattere soddisfa determinati criteri. Ad esempio, per un campo che accetta solo numeri, potremmo controllare se event.char è compreso tra '0' e '9', e prevenire l'inserimento (return "break") se non lo è. Questo approccio preventivo è più efficace rispetto alla validazione post-inserimento.

Eventi su Pulsanti della Tastiera (Shift, Ctrl)







I tasti modificatori come Shift, Ctrl e Alt sono fondamentali per implementare scorciatoie da tastiera e funzionalità avanzate nelle applicazioni. In Tkinter, possiamo rilevare lo stato di questi tasti esaminando gli attributi dell'oggetto evento passato alla funzione di callback.

L'oggetto evento fornisce informazioni sullo stato dei tasti modificatori attraverso attributi come event.state. Utilizzando operazioni di bit (bitwise operations), possiamo determinare quali tasti modificatori sono premuti. Ad esempio, è possibile verificare se il tasto Shift è premuto con (event.state & 0x0001) o se Ctrl è premuto con (event.state & 0x0004).

Una volta rilevata la combinazione di tasti desiderata, possiamo implementare funzionalità come Ctrl+S per salvare, Shift+Frecce per selezioni estese, o altre interazioni avanzate che arricchiscono l'esperienza utente dell'applicazione.