Pagina di apprendimento - Progettazione GUI di Python

Di G.D. Walters

Sommario

Introduzione

Un certo numero di anni fa ho scritto due articoli su come utilizzare Page per Full Circle Magazine (numeri 58 e 59 da febbraio e marzo 2012). A quel tempo ero molto coinvolto nell'insegnamento di Python a persone che avevano pochissimo background nella programmazione, o conoscenza intermedia della programmazione, ma erano nuove al linguaggio di programmazione Python.

Da allora, ho scritto sull'utilizzo di Python nel mondo reale controllando i dispositivi (sensori, motori, LED, ecc.) sul Raspberry Pi e interfacciandosi con il microcontrollore Arduino.

Ora, mi ritrovo a tornare ad alcune delle mie radici e lavorare su alcuni progetti Python che richiedono una GUI. Ieri sera ho parlato con mio figlio di Page e della sua capacità per un progetto su cui vuole lavorare e ho deciso di cercarlo e vedere quale fosse l'ultima versione. Per la mia gioia, è stato fino alla versione 4.9 e sarà presto la versione 4.11 per essere rilasciato all'inizio del 2018.

L'ho scaricato sul mio portatile e ho scoperto che i tutorial che ho scritto in precedenza non sono più rilevanti (in alcuni dei punti specifici) a causa di cambiamenti nel programma Pagina, così ho deciso di aggiornare i tutorial e forse venire con alcuni esempi più rilevanti per aiutare le persone lungo il loro viaggio di scoperta / riscoprire di Pagina.

Grazie a Don Rozenberg per i suoi molti anni di creazione e manutenzione del programma Page. Voglio anche ringraziare mio figlio Douglas per l'editing e il controllo della sanità mentale per questo documento. Infine, permettetemi di ringraziare un lettore di lunga data Halvard Tislavoll per tenermi in pista con molte cose che dimentico e dio per scontato.

Alcuni sfondo

Page è un progettista GUI per programmatori Python che utilizza Tkinter che supporta il set di widget Tk/ttk scritto da Don Rozenberg. Si tratta di un'estensione di Visual Tcl che produce codice Python.

Tcl è**l'acronimo di**' Tool Command Language ' ed è un linguaggio di programmazione dinamica open source. Tk è un toolkit grafico per Tcl. Tk può essere utilizzato da molti linguaggi tra cui C, Ruby, Perl, Python e Lua.

Tkinter sta per "**Tk Interface**" ed è considerato (secondo il wiki Python) come la GUI standard de facto per Python. Si tratta di un sottile strato orientato agli oggetti sopra Tcl/Tk. Mentre Tkinter non è l'unico toolkit di programmazione GUI per Python, sembra essere quello più comunemente usato.

Mentre è possibile creare GU senza un designer come Page, è un processo molto noioso e, almeno per me, piuttosto doloroso. Programmi come Page consentono uno sviluppo rapido di applicazioni o RAD, rendendo il processo di sviluppo di programmi grafici un compito abbastanza facile. È anche possibile includere l'utente finale, se disponibile, per sedersi sul processo di progettazione dell'interfaccia utente e apportare modifiche in tempo reale prima di arrivare alla parte di codifica principale.

Per chi è scritto questo

Questa guida è scritta per le persone che già conoscono le nozioni di base della programmazione Python e vogliono espandere le loro conoscenze dalle applicazioni basate su terminali per includere interfacce utente grafiche o GUI.

Perché ho scritto questa guida

Mi è sempre piaciuto insegnare, che si tratti di computer, cucina o codifica. Mi sento come se fosse la mia vera chiamata nella vita. Quando sono entrato nella programmazione informatica nel 1972, non c'era una grande quantità di informazioni disponibili per i giovani sulla programmazione, così ho afferrato a tutto quello che potevo e non ho mai smesso di cercare di imparare di più.

Python è diventato uno dei linguaggi di programmazione in più rapida crescita al mondo e negli ultimi anni è sempre stato tra i primi 5 linguaggi di programmazione a sapere. Con la sua popolarità, c'è bisogno di strumenti che consentono lo sviluppo rapido delle applicazioni per produrre interfacce user friendly oltre il terminale di comando. Page fornisce che in uno strumento gratuito che è disponibile per chiunque abbia una connessione Internet e la volontà di imparare.

Alcune Convenzioni utilizzate in questo documento

Per i comandi in una finestra del terminale, utilizzo **Liberation Mono 11pt Bold**.

Qualsiasi codice viene visualizzato in **Courier New Bold**. La dimensione del carattere cambierà di volta in volta.
Le informazioni considerate importanti saranno in **grassetto**.

Quando si descrive la navigazione nei menu, verrà utilizzato un '' per separare i vari passaggi del menu da attraversare. Un esempio di questo potrebbe **essere il menu principale Gen_Pythona0> Genera modulo di supporto**.

Requisiti

finestre

In Windows, è necessario, naturalmente, Python, sia 2.7x o 3.x. Io utilizzo Python 2.7.9.

È inoltre necessario scaricare ActiveTcl 8.6.6 (o l'ultima versione che è possibile ottenere) da ActiveState. Sto usando la versione gratuita della community.

Si vuole anche avere una sorta di editor per tenere il passo con il codice Python. Io uso un ide multipiattaforma gratuito molto bello (Ambiente di sviluppo integrato) chiamato Geany.

Linux

Mentre Linux di solito ha tutto il necessario, probabilmente vorrai usare ActiveTcl versione 8.6.6 da ActiveState. Io uso la versione gratuita della comunità.

Io uso anche (come con Windows) un IDE gratuito molto bello chiamato Geany.

Raspberry Pi (Linux)

Dovrai usare l'ultimo Raspbian Stretch. Le versioni precedenti del sistema operativo non includeva il software Tcl/Tk corretto nei repository.

Osx

Dal momento che non ho accesso a qualsiasi macchina OSX, io davvero non posso parlare a questo per esperienza diretta, ma capisco da Don che i requisiti per OSX sono gli stessi di Linux.

Installazione

finestre

Scaricare la versione più recente di ActiveTCL https://www.activestate.com/activetcl/downloads ed eseguire il file.

Molti utenti di Windows hanno un problema quando avviano La pagina. Si lamenta che il programma wish.exe non può essere trovato. Questo perché tcl non è stato installato.

Scarica la versione più recente di Page https://sourceforge.net/projects/page/ ed esegui il file. Questo

installerà Page.

Infine, creare una directory "master" sul disco rigido per contenere i file.

Linux

L'installazione in Linux richiede un po 'più di tempo rispetto a Windows, ma non terribile.

Per prima cosa si desidera scaricare ActiveTcl da ActiveState all'indirizzo https://www.activestate.com/activetcl. Ancora una volta, sto usando la versione gratuita della comunità. Il download è un archivio tar-gzipped. Una volta scaricato il file, è necessario seguire i seguenti passaggi per l'installazione.

In un terminale digitare **tar zxf /path/to/ActiveTcl-download.tar.gz**.

Passare alla cartella in cui sono stati estratti i file ed eseguire lo script del programma di installazione. Si dovrebbe fare questo come root, dal momento che l'installazione va nella cartella opt. /install.sh

Assicurarsi che la variabile PATH includa la directory che contiene i file eseguibili installati. **esportare PATH:"/opt/ActiveTcl-8.6/bin:$PATH"**

Questo è tutto per l'installazione di ActiveTcl. Ora scarica il sito di distribuzione https://sourceforge.net/projects/page/ . La pagina normalmente rileva il sistema operativo, quindi dovrebbe richiedere di scaricare l'ultima versione tar-gzipped.

Annullare il file scaricato nella home directory.

Eseguire "**./configure**" nella directory di installazione. Viene generato uno script che richiama Page.

Rimuovere tutti i file ".pagerc" nella cartella.

Raspberry Pi (Linux)

Assicurati di avere la build Raspbian Stretch più recente. È necessario installare Tcl e Tk.

In una finestra del terminale, digitare: **sudo apt-get install tcl tk**

- Al termine dell'installazione, verificarla digitando **desiderato nella** finestra del terminale. Se si apre una piccola finestra Tkinter, questo è fatto.

- Scaricare la distribuzione della Pagina da https://sourceforge.net/projects/page/
- Una volta scaricato, rimuovere il file scaricato nella home directory.

- Eseguire **./configure** nella cartella di installazione page. Verrà generato lo script che richiamerà Page.

- Utilizzare l'editor di menu (Menu principale Proprietà Preferences (Preferenze) Editor menu principale). Fare clic sul pulsante

  Gruppo di programmazione a sinistra e quindi il pulsante Nuovo elemento a destra. Digitare Pagina nella casella Digitazione nome. Quindi utilizzare il pulsante Sfoglia per trovare il file 'pagina' nella cartella di installazione della pagina. È possibile impostare un'icona, se lo si desidera, facendo clic sull'immagine a sinistra delle proprietà di avvio, passare alla cartella di installazione pagina, e sotto **la cartella delle icone di** pagina, selezionare un'icona che ti piace. Io uso **'page32.png'**.

- Infine fare clic **sul pulsante OK.** Ora è possibile testare l'installazione.

Osx

Ancora una volta, da quello che Don mi dice, i passaggi per l'installazione sono più o meno gli stessi di Linux.

Lascio l'installazione di Geany o qualsiasi IDE si sceglie di utilizzare per voi.

Come in Windows, creare una "directory master" per contenere i file di sviluppo.

Un'ultima cosa sui programmi IDE. Non sono creati uguali. Mentre ci sono decine là fuori, alcuni hanno nascosto "opzioni" che potrebbero rendere il vostro lavoro più difficile piuttosto che più facile. Mentre questa è strettamente un'opinione personale, suggerirei di non utilizzare IDLE come editor di scelta. C'è una "caratteristica" che avvia Python con il -i per entrare in modalità interattiva dopo che il programma è terminato. Questo lascia l'interfaccia utente sullo schermo e potrebbe causare qualche preoccupazione (Grazie Halvard per aver sottolineato questo a me). Anche in questo caso, la scelta di un IDE è la tua scelta. Mi piace Geany, altri come Sublime Text mentre altri come IDLE.

Capitolo 1 - Guida introduttiva

Per avviare Pagina, fare clic sull'icona. Dopo alcuni secondi, dovresti essere presentato con tre (almeno) nuove finestre sull'area di lavoro desktop. Dovrebbe assomigliare a questo...

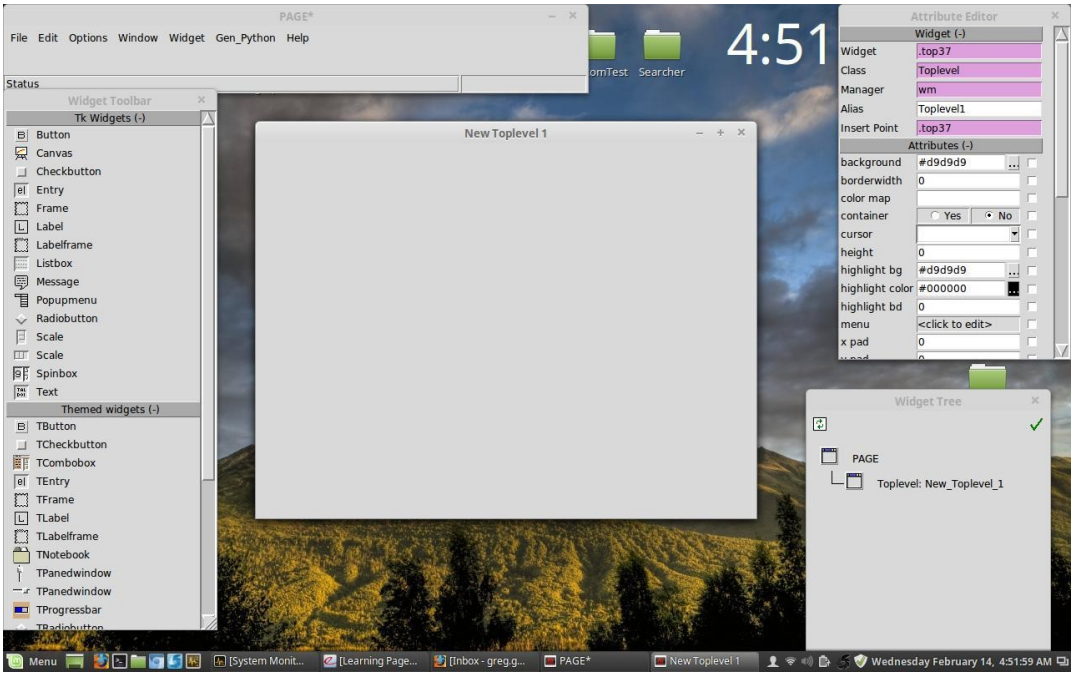

Immagine 1: Le 5 finestre della pagina principale

In senso orario dall'alto a sinistra, sono la finestra principale, l'Editor attributi, Widget Tree e la barra degli strumenti Widget. Il widget Livello superiore viene aggiunto all'avvio della pagina (a partire dalla versione 4.10). Qui di seguito, puoi vederli ciascuno.

Immagine 2: La finestra principale della pagina

La finestra principale contiene il file standard (Apri Salva, ecc.) così come le opzioni per affrontare le varie finestre, generazione Python e altro ancora. Riducendo al minimo questa finestra si ridurranno a icona anche tutte le altre finestre della Pagina, ad eccezione del widget Livello superiore.

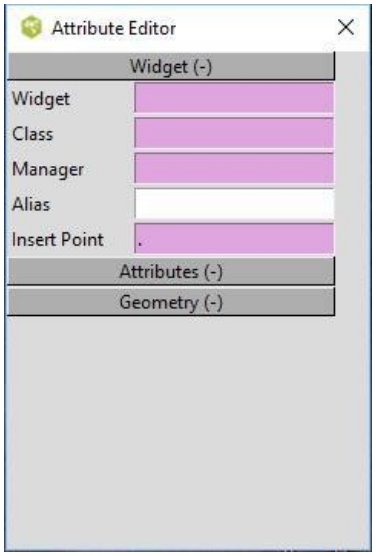

Immagine 3: Editor attributi

La finestra Editor attributi sarà una delle sezioni più utilizzate durante le sessioni di creazione della GUI. Qui si impostano le parti visive importanti dei vari widget nel vostro progetto come posizione, dimensione, colore, testo, alias (il nome del widget nel codice) e altro ancora.

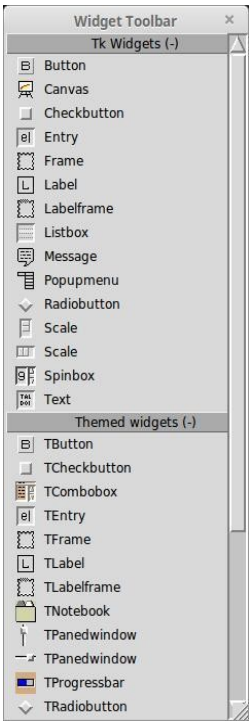

Immagine 4: Barra degli strumenti widget

La casella degli strumenti contiene tutti i widget che verranno utilizzati per progettare la GUI. Questi widget includono elementi come oggetti di testo statici come etichette, oggetti di immissione di testo, pulsanti tra cui pulsanti di controllo e di opzione, caselle combinate e molto altro ancora.

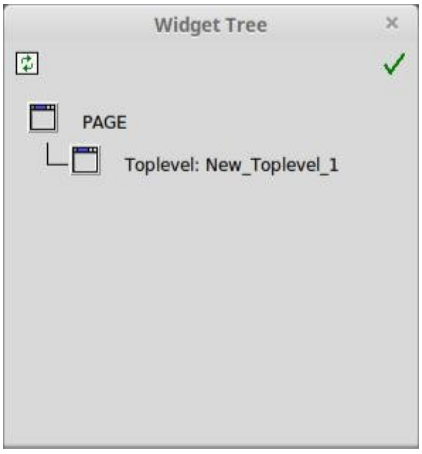

Immagine 5: L'albero dei widget

L'albero widget mostra tutti i widget in una visualizzazione gerarchica. È possibile fare clic su uno qualsiasi dei widget per selezionarli nella finestra di progettazione principale o per modificare gli attributi. È anche possibile fare clic destro su un widget qui per selezionare altre opzioni come rilegatura.

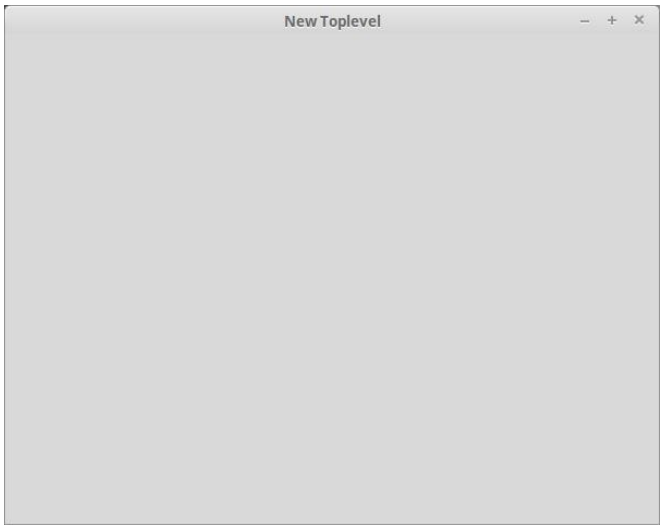

Immagine 6: Nuovo widget di livello superiore

Infine, a partire dalla pagina 4.10, un nuovo widget di livello superiore verrà presentato automaticamente quando si avvia Pagina senza alcun parametro della riga di comando. Se si desidera rielaborare un progetto precedente, è sufficiente avviare Pagina con il file .tcl del progetto (insieme al percorso).

Il nostro primo programma

Il nostro primo programma includerà semplicemente una finestra GUI con un singolo pulsante che chiude e distrugge la finestra.

La finestra principale - Livello superiore

Per avviare un nuovo progetto, è sufficiente avviare Page. Questo aprirà un form vuoto (insieme alle altre finestre necessarie) che sarà il form principale per la tua applicazione. Sarà qualcosa di simile a questo...

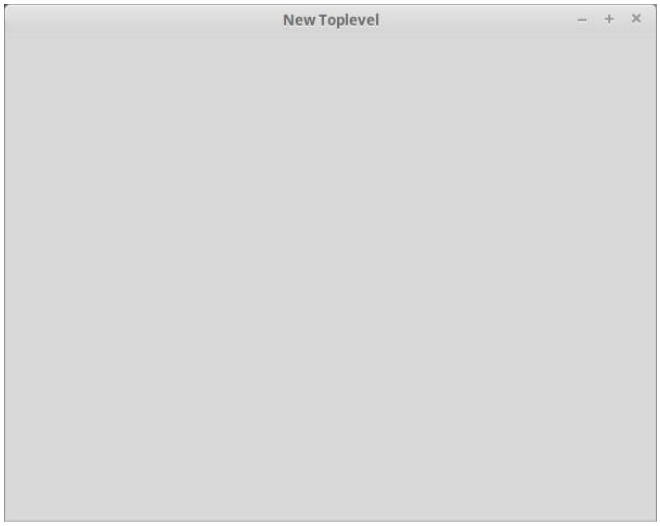

Immagine 7: Il widget di primo livello all'avvio

Considera questo la tela bianca che terrà il tuo capolavoro. Può essere spostato sullo schermo e/o ridimensionato per adattarlo ai tuoi scopi. Dopo aver posizionato il punto in cui si desidera visualizzare la finestra principale all'avvio dell'app, è necessario modificare il titolo. Per impostazione predefinita, è intitolato "Nuovo livello superiore". Cambiamola in "My First Page App". A tale scopo, si utilizzerà l'Editor attributi. Utilizzare la barra di scorrimento a destra dell'Editor attributi, scorrere verso il basso fino alla riga '**title**' che si trova appena sopra la barra Geometria. Modificare il testo presente da "Nuovo livello superiore" a "App La mia prima pagina". Premendo il tasto [Invio] sulla tastiera, si "imposta" la modifica. Scorrere ora verso l'alto fino alla parte superiore della finestra Editor attributi. Ci vedrete cinque opzioni nella sezione 'Widget'. L'unico che è necessario modificare in questa sezione è quello contrassegnato come '**Alias**'. Questo è ciò che chiameremo il modulo / widget nel nostro programma per far facilmente riferimento ad esso. Assicurati di usare qualcosa di descrittivo, ma in una sola parola. Per i nostri scopi di apprendimento, lo chiameremo "MainForm". Immetterlo nella casella di testo accanto alla parola "Alias".

Si potrebbe notare che è spuntata una nuova finestra che è denominata "Widget Tree". Verrà mostrata una rappresentazione gerarchica del modulo e di tutti i widget ad esso associati. Per ora, puoi semplicemente fare clic sul segno di spunta per chiuderlo, oppure puoi spostarlo da qualche parte. Se mai si desidera vedere di nuovo e non riesco a trovarlo, è possibile arrivare ad esso dalla **finestra principale . Proprietà Window . Widget Tree** o semplicemente premendo <Alt>W.

Prima di andare oltre, si dovrebbe salvare il vostro lavoro. Ti ti tirerai spesso a farlo, specialmente mentre stai imparando. Nella finestra principale, selezionare **File Salva**. Verrà richiesto di selezionare la cartella e il nome del progetto. Per ora, utilizzare "FirstApp" come nome del file. Si potrebbe anche voler metterlo in un

cartella separata per organizzare i file. Si noti inoltre che questo file verrà salvato con un'estensione ".tcl". Questo è il modo in cui dovrebbe essere, dal momento che stiamo creando lo script Tcl per la GUI. Non ci sarà alcun codice Python per un po '. Questo file consente di rielaborare la GUI in un secondo momento.

Aggiunta di un pulsante

Ora, mettiamo un semplice pulsante da qualche parte vicino al centro del nostro modulo che uscirà dal modulo.

Sulla barra degli strumenti widget verrà visualizzato il pulsante "**nella**parte superiore della casella degli strumenti. Fare clic sul pulsante "**",**quindi fare clic in un punto qualsiasi del modulo principale.

Il pulsante che abbiamo appena creato avrà otto quadrati neri che lo circondano. Si tratta di maniglie che è possibile utilizzare per ridimensionare il widget. È inoltre possibile fare clic e trascinare il widget nel punto in cui si desidera che si trovi all'interno del modulo.

Una volta che abbiamo posto il widget del pulsante, vai all'Editor attributi e imposta l'alias su "**btnExit**". Scorrere verso il basso fino a trovare **l'attributo** di testo e modificarlo in "**Esci**". Non appena si preme entrare, si vedrà l'etichetta sul pulsante nel nostro modulo cambiare da "Pulsante" a "Esci". Ora utilizzare le maniglie sul widget pulsante per renderlo un po 'più ampio. Una volta che hai nel modo che ti piace, salvare il file .tcl di nuovo per mantenerlo aggiornato.

Abbiamo un certo numero di cose da fare prima di essere fatto, ma lo faremo tra un attimo. Per ora, possiamo generare il nostro codice Python per vedere come funziona tutto.
Nel menu principale della finestra della Pagina, si farà clic **su Gen_Python Generare la voce di** menu Gui di Python. Dopo un paio di secondi, una nuova finestra verrà visualizzata con il nostro codice Python di base in un semplice editor. Sentiti libero di scorrerlo per vedere come appare il codice, ma non cambiare nulla. Successivamente, premere il **pulsante** Salva, che salverà il codice Python generato da Page per noi.

Si noterà che c'è **un pulsante** Esegui in questa finestra. Se si fa clic su di esso ora, verrà visualizzata una finestra di messaggio che informa che si è verificato un errore con il messaggio "Nessun*modulo di supporto è stato creato e salvato*. " Va bene, dal momento che è quello che faremo dopo. Fare clic **sul** pulsante Ok nella finestra di messaggio, quindi fare clic sul **pulsante** Chiudi nella finestra Python generato.

Torna nella finestra Pagina principale, fare di nuovo clic **su Gen_Python Genera modulo di supporto**. Questo si aprirà di nuovo la finestra Python generato, questa volta con il resto del codice che sarà necessario per eseguire la nostra applicazione. Anche in questo caso, sentitevi liberi di scorrere questo file e quando avete finito fare clic su **Salva**. Ora fare clic sul **pulsante** Esegui. In pochi secondi, vedrai la finestra Prima app. Dovrebbe assomigliare a questo.

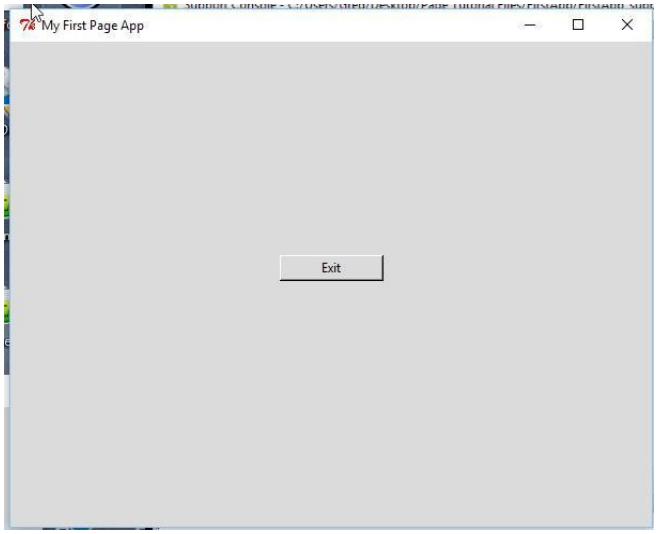

Immagine 8: La nostra prima pagina App

Sentitevi liberi di fare clic sul **pulsante** Esci, ma non accadrà ancora nulla, perché non abbiamo scritto alcun codice per gestire il clic del mouse. Va bene. Lo faremo in un attimo. Per il momento, chiudere la finestra demo (la "X" in alto a destra della finestra) e chiudere l'editor di codice.

Ci si potrebbe chiedere perché c'è bisogno di un modulo di supporto che è separato dal file principale. Questa operazione viene eseguita in modo che il file Python principale (in questo caso "FirstApp.py") viene mantenuto solo per le definizioni necessarie per creare l'interfaccia utente. Tutte le altre funzioni, come il codice che controlla ciò che accade quando si fa clic sul pulsante di uscita, vengono mantenute nel modulo di supporto (in questo caso denominato "FirstApp_support.py"). In questo modo è facile mantenere il codice senza modificare accidentalmente il file di definizione dell'interfaccia utente ed è possibile modificare o modificare l'interfaccia utente senza dover copiare e incollare il codice da una copia di sicurezza. A proposito, ogni volta che salvi i tuoi file in Page, fa un backup del file precedente, in modo da poter sempre tornare indietro per vedere cosa avevi.

Suggerimento: *In genere, si desidera generare il modulo di supporto solo quando si ha finito con la progettazione GUI. C'è la possibilità che il codice che hai inserito nel modulo di supporto potrebbe ottenere strapazzate dal nuovo codice che Page mette in. Gli errori saranno piccoli e di solito richiedono semplicemente tagliare e incollare di poche righe di codice per riorganizzare le cose. Don ha fatto un ottimo lavoro per assicurarsi che non ci siano grandi ostacoli qui.*

Per far funzionare il pulsante Esci, è necessario **binding**creare l'associazione. Questo crea il codice che collega il widget, in questo caso il pulsante Exit, a una funzione che chiamerà la routine sys.exit() per uscire ed eliminare questa finestra.

Per creare l'associazione, fare clic con il pulsante destro del mouse su btnExit nella finestra di progettazione **e selezionare Associazioni...** dal menu a comparsa. Si aprirà un'altra finestra che semplifica la creazione di vari tipi di relazioni

tra gli eventi widget e il nostro codice. In questo caso, vogliamo assicurarci che **btnExit sia** evidenziato quindi premere il **pulsante Aggiungi** (pulsante all'estrema sinistra) sulla barra degli strumenti. Vedrai un menu pop-up che assomiglia all'immagine qui sotto.

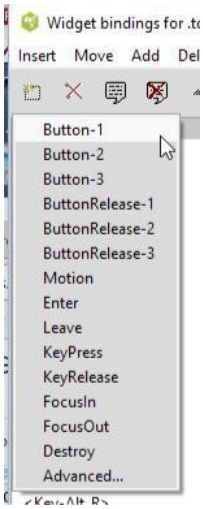

Immagine 9: Opzioni di rilegatura

Ci sono un certo numero di opzioni qui, e ci stiamo andando a selezionare **Button-1**. Ciò equivale al pulsante sinistro del mouse. Button-2 è il pulsante centrale (se il mouse ha un pulsante centrale) e Button-3 è il pulsante destro. Ci sono una serie di altre opzioni, alcune delle quali legano altri eventi come quelli che sparano sul rilascio del pulsante, preme i tasti e altro ancora. Ne esploreremo alcuni più avanti in questo documento.

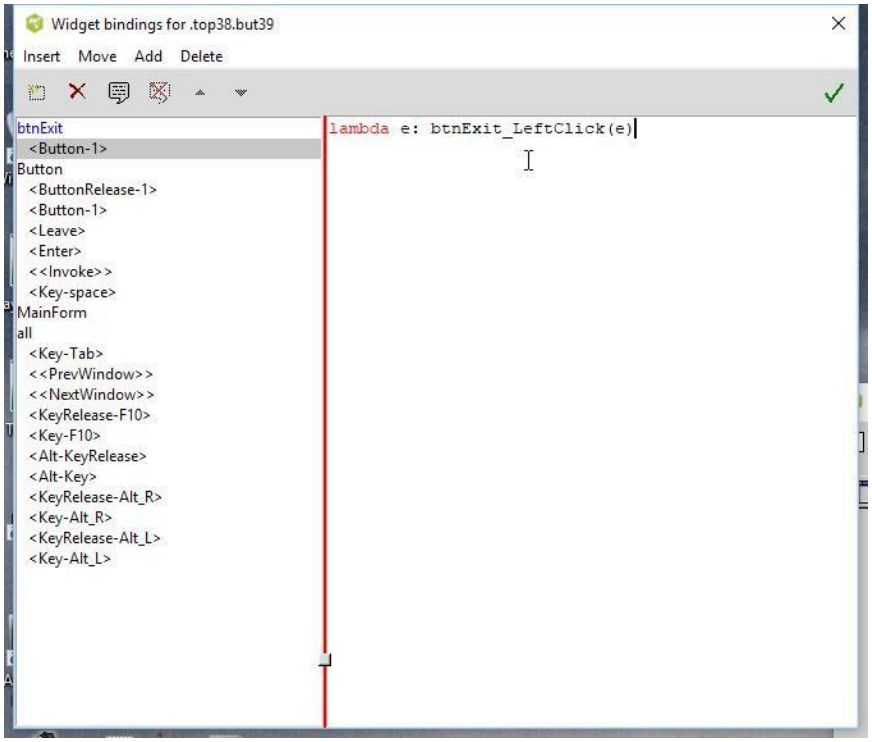

Immagine 10: La finestra di rilegatura

Una volta selezionato Button-1, dovrebbe essere visualizzato "lambda e: xxx(e)" nel pannello laterale destro. Questa è la nostra linea di legame. Tuttavia, dobbiamo cambiare un po' questo per renderlo un po' più auto-documentato. Cambieremo il "**xxx(e)**" in "**btnExit_LeftClick(e)**". Evidenziare "xxx" e sostituirlo con "**btnExit_LeftClick**".

A questo punto, salvare il progetto, Generare il codice Python e quindi il file di supporto. Quando si genera il file di supporto, verrà visualizzata una finestra di messaggio che indica che il file di supporto esiste già, insieme a tre opzioni. Uno è quello di sostituire il file, uno per utilizzare il file esistente e uno per aggiornare il file. A questo punto, non abbiamo fatto nulla nel file esistente, quindi non sarebbe una grande perdita se lo sostituiamo con una nuova versione. Tuttavia, una volta che si ottiene facendo di più, la sostituzione del file vi farà perdere eventuali modifiche o codice personalizzato che avete già inserito. Quindi, per essere sicuri, selezionare Aggiorna.

Ora, se si confronta il FirstApp.py originale con quello nuovo appena generato, si vedrà che è stata generata una riga aggiuntiva che contiene la nostra associazione nella parte inferiore della definizione del pulsante. Lo è...

```
self.btnExit.bind('<Button1>',lambda e:FirstApp_support.btnExit_LeftClick(e))
```

Suggerimento: è possibile notare che nella riga di codice precedente viene chiamato *il parametro passato alla routine del gestore eventi _LeftClick . Nel codice generato da Page riportato di seguito nel file di supporto, viene chiamato (p1). Questo perché l'associazione di un evento passa informazioni sull'evento che include un*

quantità di informazioni alla funzione di gestione. Se non è necessario passare alcun parametro, è possibile utilizzare l'attributo di comando del pulsante o di un altro widget.

Il file di supporto ha anche la nuova routine btnExit_LeftClick(p1) già generata per noi.

```
def btnExit_LeftClick(p1):
    print('FirstApp_support.btnExit_LeftClick')
    sys.stdout.flush()
```

Se non aggiungi altro codice, quando esegui l'app ora, il terminale mostrerà

"FirstApp_support.btnExit_LeftClick" ogni volta che fai clic sul pulsante Esci.

Ora per rendere il pulsante effettivamente funzionare nel modo in cui vogliamo, che è quello di chiudere questa finestra, abbiamo bisogno di aggiungere quanto segue alla routine btnExit_LeftClick nel file di supporto. Page fornisce una funzione chiamata destroy_window(), che è il modo corretto per terminare il nostro programma GUI. Chiameremo quel codice da qui.

```
destroy_window()
```

Ora la routine dovrebbe essere così.

```
def btnExit_LeftClick(p1):
    print('FirstApp_support.btnExit_LeftClick')
    sys.stdout.flush()
    destroy_window()
```

Ora, quando si esegue il programma, facendo clic sul pulsante Esci causerà la chiusura della finestra.

Ora si può vedere che è abbastanza facile creare programmi GUI per Python utilizzando Page. Successivamente creeremo un programma più complicato che dimostra alcuni dei vari widget disponibili all'interno di Page.

Capitolo 2 - Andare avanti

Il nostro prossimo progetto creerà una finestra con i seguenti widget...

Pulsanti

Cornici

Etichette

Pulsante di controllo Casella di testo

(widget Immissione riga singola)

Pulsante di opzione

Questo semplice programma eseguirà le seguenti attività..:

Fotogrammi: illustra i widget di raggruppamento in set visivi logici.

widget di immissione: dimostra un modo per l'utente di fornire dati che possono essere recuperati a livello di codice, in questo caso un pulsante. Facendo clic sul pulsante si ottiene le informazioni dal widget di immissione e le si stampa nella finestra del terminale.

Pulsanti di opzione: viene illustrata la funzione di raggruppamento dei pulsanti di opzione e l'impostazione dinamica di un'etichetta con testo.

Pulsanti di controllo: dimostra la funzione On/Off dei pulsanti di controllo e stampa nella finestra del terminale il valore dei widget selezionati o su quando si fa clic sul pulsante associato.

Widget Etichetta: dimostra la possibilità di modificare dinamicamente il testo visualizzato all'utente.

Prima di iniziare, dovremmo esplorare ciò che ognuno di questi widget fa.

Pulsanti

I pulsanti, come abbiamo scoperto nel nostro primo esempio di programma, sono widget che quando cliccato (premuto con un pulsante del mouse o una pressione del tasto della tastiera o anche toccato con un dito su un touch screen) genera un evento che il nostro programma agirà su. Tutti abbiamo familiarità con i pulsanti.

Fotogrammi

I frame forniscono un modo semplice per raggruppare gli elementi che appartengono logicamente insieme. Un Labelframe è semplicemente un fotogramma con un'etichetta allegata che fornisce un'indicazione immediata sull'intento di tale raggruppamento.

Etichette

Le etichette sono in genere testo statico che mostra all'utente finale a cosa a cui è a scopo un particolare widget. È statico poiché non cambia mai (o cambia raramente) durante la vita del programma. Ci sono momenti in cui potresti voler usare un'etichetta come display dinamico per mostrare ciò che sta accadendo da qualche parte nel programma (che è quello che faremo con il progetto in questo capitolo).

Caselle di testo per l'immissione a riga singola

Una casella di immissione è un widget che consente all'utente finale di digitare informazioni o dati. Ci sono due diversi tipi di widget di immissione di testo nella libreria di widget Tkinter standard, un widget di immissione e un widget di testo. Utilizzare il widget di immissione per l'immissione di dati a riga singola come nome, cognome, indirizzo, ecc. Quando hai bisogno di voci di dati su più linee, come una serie di istruzioni, usa il widget di testo.

Pulsanti di controllo

Le caselle di controllo forniscono un riferimento visivo allo stato Sì/No o On/Off di una variabile o di un'opzione. Le caselle di controllo sono autonome e normalmente non interagiscono con altri widget. Considerarli come uno strumento visivo molti-di-molti. Molte delle caselle di controllo possono essere selezionate o selezionate contemporaneamente.

Pulsanti

I pulsanti di opzione, ad esempio una casella di controllo, forniscono un riferimento visivo allo stato Sì/No o On/Off, ma in genere fanno parte del set di widget Uno-Di-Molti. È possibile selezionare uno stato Di scelta (On) all'interno di un gruppo alla volta e raggrupparsi all'interno di un frame. Quando è selezionato un pulsante di opzione, tutti gli altri membri del gruppo vengono selezionati. Tutti i pulsanti di opzione in un gruppo utilizzano lo stesso nome di variabile. L'attributo value verrà utilizzato per specificare cosa verrà inviato da ogni pulsante a tale variabile.

Inizio del progetto - Layout

È sempre bene avere un'idea dell'aspetto dell'interfaccia utente prima di iniziare. Mi piace disegnare il mio disegno su carta prima, solo per avere un'idea di ciò di cui ho bisogno in modo di spazio e dimensioni. Ecco come sarà il nostro progetto finito nel nostro Page designer...

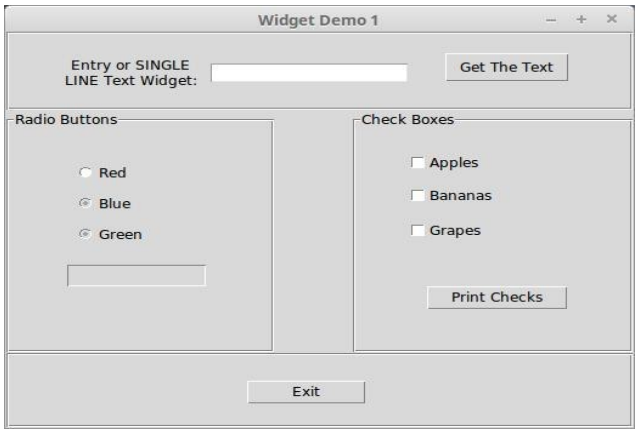

Immagine 11: Seconda app

Come potete vedere, abbiamo bisogno di quattro fotogrammi, due dei quali sono Labelframe e due che sono fotogrammi regolari. Avremo anche bisogno di tre pulsanti standard, tre pulsanti di opzione, tre pulsanti di controllo, un widget di immissione di una singola riga e due etichette. È possibile che non si riconosca immediatamente il secondo widget etichetta. È il rettangolo affondato a sinistra. Non ha etichetta in questo momento, dal momento che il testo verrà generato dinamicamente dalla selezione dei pulsanti di opzione. Eremo impostare la nostra GUI prima con tutti i widget e i loro attributi, poi impostaremo le nostre associazioni e infine scrivere Capitolo 2 - Andare avantiil codice richiesto.

Pagina iniziale e spostare il widget Nuovo livello superiore in modo che sia centrato sullo schermo e salvare il progetto immediatamente. Chiamalo "SecondApp".

Ti darò un elenco di attributi per ciascuno dei widget che usiamo, che devono essere impostati nell'Editor attributi. Il valore da utilizzare è in grassetto. La posizione X, la posizione Y, la larghezza e l'altezza vengono impostate nella sezione Geometria dell'Editor. Ecco il valore impostato per il nostro widget finestra toplevel.

alias: **livello superiore**
titolo: **Widget Demo 1**
Posizione x: **344**
posizione: **162**
larghezza: **517**
altezza: **386**

Successivamente, abbiamo bisogno del nostro primo fotogramma. Fare clic sul pulsante della cornice nella casella degli strumenti e posizionarlo in un punto qualsiasi

widget di primo livello. La posizione, l'alias e la dimensione verranno impostati nell'editor di attributi con le seguenti informazioni:

alias: **frameTop**
Posizione x: **1**
Posizione y: **0**
larghezza: **515**
altezza: **76**

Suggerimento: Oltre a inserire tutti gli attributi *nell'Editor attributi, è possibile fare clic con il pulsante destro del mouse sul widget e impostare* ***rapidamente*** alcuni degli attributi più importanti da un menu *a comparsa. Questo include l'Alias e altri attributi specifici del widget come il testo per un widget pulsante. È possibile eseguire la stessa operazione dalla finestra Albero widget.*

Ora popoleremo il telaio superiore con un widget Etichetta standard, un widget Entry e un

widget Pulsante. Iniziare con il widget Etichetta e posizionarlo da qualche parte nella parte sinistra del fotogramma superiore.
Impostare i relativi attributi sui seguenti valori:

alias: **Label1**
testo: **widget di testo di immissione o SINGLE LINE:**
width: **116** *(diverso dalla larghezza nella sezione Geometria)*
lunghezza del ritorno a **capo: 110**
Posizione x: **30**
posizione: **20**
larghezza: **146**
altezza: **41**

Nota: *Quando si imposta la lunghezza del ritorno a capo di un widget (per i widget che lo supportano), si utilizza un valore di unità dello schermo (pixel). Potrebbe essere necessario modificare questo valore per farlo apparire esattamente nel modo desiderato.*
Posizionare quindi un widget Voce al centro del frame e impostare gli attributi come segue:

alias: **entryExample**
Posizione x: **167**
posizione: **30**
larghezza: **164**
altezza: **20**

Infine, posizionate un pulsante standard vicino al lato destro del telaio. Gli attributi devono essere:

alias: **btnGetText**
testo:ottenere il testo
Posizione x: **360**
posizione: **30**

larghezza: **103**
altezza: **29**

Nota: *La larghezza e l'altezza sono dinamiche in questo caso e già impostate per noi quando abbiamo inserito il testo per il pulsante. È sempre possibile impostare manualmente i valori in base alle proprie esigenze.*

Ora ci occuperemo del lato sinistro della finestra che terrà i pulsanti di opzione. Inizieremo posizionando un widget Labelframe vicino al centro della finestra Livello superiore. Una volta fatto questo, impostare i suoi attributi come segue:

alias: **lframeRadioButtons**
testo:pulsanti **di opzione**
Posizione x: **1**
posizione: **77**
larghezza: **220**
altezza: **235**

Suggerimento: Se si imposta accidentalmente un valore nelle caselle *di immissione x o y relative relative, le cose probabilmente non avranno l'aspetto previsto (come la scomparsa del widget). In questo caso, è sufficiente inserire uno "0" (zero) nella casella di immissione che hai cambiato e le cose andranno alla normalità.*

Successivamente abbiamo bisogno di inserire tre widget RadioButton nel frame di sinistra. Sentitevi liberi di metterli tutti in una sola volta e poi tornare indietro e impostare gli attributi. In questo caso, è possibile utilizzare l'albero dei widget per selezionare il pulsante con cui lavorare anziché fare clic su ogni widget nella finestra di progettazione. Assicurati di impostare tutti gli attributi che vi faccio vedere.

alias: **rbRed**
testo: **Rosso**
valore: **Rosso**
variabile: **colori**
Posizione x: **50**
posizione: **50**

alias: **rbBlue**
testo: **Blu**
valore: **Blu**
variabile: **colori**
Posizione x: **50**
posizione: **80**

alias: **rbGreen**
testo: **Verde**
valore: **Verde**
variabile: **colori**
Posizione x: **50**

Posizione y: **110**

L'ultima cosa che dobbiamo inserire in questo Labelframe è l'etichetta che mostra il valore di stato del pulsante di opzione selezionato. Come al solito, il posto è da qualche parte nella cornice sinistra che è vuoto.

alias: **lblRbInfo**
rilievo: **affondato**
variabile di **testo:colori**
Posizione x: **50**
posizione: **150**
larghezza: **114**
altezza: **21**

Successivamente, è necessario un altro Labelframe sul lato destro dell'interfaccia utente. Questo terrà i nostri pulsanti di controllo e un pulsante standard.

alias: **lframeCheckBoxes**
testo:caselle **di controllo**
Posizione x: **286**
posizione: **77**
larghezza: **230**
altezza: **235**

Ora aggiungere tre pulsanti di controllo e impostare gli attributi come segue.

alias: **chkApples**
testo: **Mele**
variabile: **che39**
Posizione x: **40**
posizione: **40**

Pulsante di controllo #2

alias: **chkBananas**
testo: **Banane**
variabile: **che40**
Posizione x: **40**
posizione: **72**

Pulsante di controllo #3

alias: **chkGrapes**
testo: **Uva**

variabile: **che41**
Posizione x: **40**
posizione: **106**

Infine aggiungere il pulsante standard. Quando si fa clic su questo pulsante chiamerà del codice che stamperà quali pulsanti, se presenti, sono selezionati.

alias: **btnPrintChecks**
testo:stampare assegni
Posizione x: **60**
posizione: **170**
larghezza: **117**
altezza: **24**

L'ultima cosa che dobbiamo fare nel nostro processo di layout è quello di mettere un altro telaio standard nella parte inferiore del nostro modulo. Questo terrà un pulsante standard per chiudere l'applicazione.

alias: **frameBottom**
Posizione x: **1**
Posizione y: **313**
larghezza: **515**
altezza: **72**

Ultimo ma non meno importante, posizionare il pulsante standard vicino al centro del frame inferiore.

alias: **btnExit**
testo: **Esci**
Posizione x: **197**
posizione: **26**
larghezza: **99**
altezza: **24**

Ora assicurati di salvare il tuo progetto e generare il codice Python e, se lo desideri, il modulo di supporto. In questo modo, possiamo passare alle associazioni dei nostri widget alle routine di codice.

Fase 2 - Attacchi

Fortunatamente, la maggior parte dei nostri attacchi sono già fatti per noi, grazie a Page. Ma ci sono alcune cose che hanno ancora bisogno di un po 'di attenzione.

Gli unici attacchi che dobbiamo affrontare saranno i nostri pulsanti standard. Inizieremo con la parte superiore più

, **btnGetText**. Fare clic con il pulsante destro del mouse su di esso, facendo attenzione a non spostare il pulsante all'interno del frame, e selezionare '**Associazioni**'. In alternativa, è possibile fare clic con il pulsante destro del mouse sulla relativa voce nella finestra Albero widget

Vedrai il nostro widget elencato nella parte superiore del pannello a sinistra. Vogliamo associare un evento clic con il pulsante sinistro, quindi fare clic sul testo '**btnGetText**', quindi fare clic sul pulsante **Aggiungi** sulla barra dei menu (il pulsante all'estrema sinistra). Verrà visualizzata una finestra popup e verrà selezionato **<Button-1>**.

Sul lato destro della finestra delle associazioni verrà visualizzato il codice di associazione "**lambda e: xxx(e)**". Verrà modificato il '**xxx(e)**' parte in '**btnGetText_lclick(e)**' e quindi fare clic sul segno di spunta a destra della finestra per chiuderlo.

Faremo la stessa cosa con il pulsante '**Print Checks**', ma puntaremo a una funzione chiamata

btnPrintChecks_lclick(e).

Infine, associare la funzione '**btnExit_lclick(e)**' a **btnExit**.

Ancora una volta, salvare il file .tcl e generare il codice Python e il modulo di supporto.

Fase 3 – Il codice

Ora, scriveremo il codice che controllierà ciò che accade quando vengono utilizzati i nostri widget.

Inizia aprendo il modulo SecondApp_support.py nel tuo editor preferito. Vedrai il codice di importazione Python standard, che ignoreremo per ora. La prima cosa che eremo esaminare (ma non cambiare) è la routine **set_Tk_var()**. Come tutto il resto di questo file è stato generato per noi da Page.

```
def set_Tk_var():
    Colori globali
    Colori : StringVar()
    globale che39
    che39 - StringVar()
    globale che40
    che40 - StringVar()
    globale che41
    che41 - StringVar()
```

Nel codice precedente sono presenti quattro variabili globali definite e impostate sul tipo **StringVar**. Il primo, Colors, è la variabile condivisa che associa i RadioButtons insieme come gruppo. Viene impostato automaticamente quando si fa clic su uno dei radioButton. Per visualizzare o utilizzare il valore all'interno della variabile, utilizzate **il metodo .get().** In genere, si imposta anche uno dei RadioButton all'avvio del programma come valore predefinito utilizzando il metodo

.set(). Gli altri tre sono per i nostri widget Pulsante Casella di controllo.

Le altre tre routine richiederanno un po 'di attenzione.

```
def btnExit_lclick(p1):
    print('SecondApp_support.btnExit_lclick')
    sys.stdout.flush()
```

La prima è la routine che viene chiamata quando il mouse causa la btnExit_lclick'evento. Vogliamo che il programma si interda quando questo accade (proprio come nel primo programma di esempio) quindi dopo la riga `sys.stdout.flush()`, inserire `destroy_window()`. Ora dovrebbe essere simile a questo

```
def btnExit_lclick(p1):
    print('SecondApp_support.btnExit_lclick')
    sys.stdout.flush()
    destroy_window()

def btnGetText_lclick(p1):
    print('SecondApp_support.btnGetText_lclick')
    sys.stdout.flush()
```

La seconda routine stamperà nella finestra del terminale tutto ciò che è stato digitato nel widget di immissione. Quindi, di nuovo

dopo la riga `sys.stdout.flush()` digitare `print(w.entryExample.get())`. Ottiene il valore della voce

Widget. Usiamo il **w. per fare riferimento** in modo esplicito all'oggetto **w** creato nella routine **init che** ha creato un'istanza dell'interfaccia utente.

```
def btnGetText_lclick(p1):
    print('SecondApp_support.btnGetText_lclick')
    sys.stdout.flush()
    print(w.entryExample.get())

def btnPrintChecks_lclick(p1):
    print('SecondApp_support.btnPrintChecks_lclick')
    sys.stdout.flush()
```

Ultimo, ma non meno importante, vogliamo stampare quale dei pulsanti di casella di controllo sono selezionati (selezionati) alla finestra del terminale. Questa volta, avremo un certo numero di righe di codice da inserire...

```
    if che39.get() : "1": due segni di uguale su ciascuna delle istruzioni 'if' print("chkApples")
    if che40.get() : "1":
        print("chkBananas")
            if che41.get() - "1":
            print("chkGrapes")
```

Ci sono altre due cose che devono essere fatte. In primo luogo, dobbiamo assicurarci che all'avvio del programma, nessuno dei pulsanti di controllo sia selezionato. Useremo una semplice routine per eseguire questa operazione.

```
    def ClearChecks():

        che41.set("0")
```

```
che40.set("0")
che39.set("0")
```

Ho messo che subito dopo la routine **set_Tk_var(),** ma si può mettere praticamente ovunque prima della riga che legge se __name__ **'__main__':**

Infine, è necessario chiamare il set uno dei RadioButtons come impostazione predefinita all'avvio del programma. Lo faremo e chiameremo la **funzione ClearChecks** dalla routine **init.** Subito dopo la riga che dice radice - in **alto**, inserire:

```
Colors.set("Blu")
ClearChecks()
```

Inserendolo alla fine della routine **init,** queste righe di codice verranno eseguite DOPO la creazione della GUI, ma prima che venga visualizzata.

A questo punto, se si utilizza una versione di Page precedente alla 4.11, spostarsi all'inizio del file e aggiungere il codice seguente alla seconda riga:

```
#    -- codifica: utf-8 -
```

Ciò contribuirà a garantire che i caratteri Unicode vengano gestiti se si immette qualsiasi nel widget di immissione. E 'una buona abitudine di entrare in, uno che devo ammettere, dimenticare il più delle volte. (Grazie Halvard per tenermi sulla strada giusta!) Nella versione 4.11, Don ha cambiato Pagina per eseguire automaticamente questa operazione per noi.

Salvare il programma ed eseguirlo per vedere come hai fatto. Il pulsante di opzione "Blu" deve essere selezionato e la parola "Blu" deve essere nell'etichetta affondata. Inoltre, tutte le caselle di controllo devono essere deselezionate. Digitare qualcosa nella casella di immissione e fare clic sul pulsante "Ottieni il testo". Qualunque cosa hai digitato nel widget di immissione dovrebbe essere stampata nella finestra del terminale che hai usato per avviare il programma. Selezionare uno o più pulsanti della casella di controllo e fare clic sul pulsante "Stampa assegni". Il testo associato a tale pulsante dovrebbe essere visualizzato nella finestra del terminale.

Ottimo lavoro.

Capitolo 3 – Un progetto più realistico

Come ho detto prima, quando ho iniziato a utilizzare Page, ho scritto un tutorial per Full Circle Magazine, che Don è stato così gentile da rendere disponibile dal suo sito web. Era il 2012. Ora che il 2018 è qui e Page è maturata notevolmente, ho deciso di rivisitare quel progetto e aggiornarlo un po'.
Lo scopo di questo progetto è quello di creare un cercatore di file che troverà file audio e video per le loro estensioni che l'utente può selezionare da una serie di pulsanti di controllo, quindi visualizzare i risultati in un foglio di calcolo come griglia. Questo dimostrerà:

Finestre di dialogo pop-up, in particolare la finestra di dialogo
Selezione directory Uso corretto dei pulsanti Di controllo

Utilizzo del widget Visualizzazione struttura a scorrimento

La possibilità di cambiare il cursore in un cursore "occupato" e indietro. Riempimento dinamico di un widget di immissione

Questa volta, non ti darò la posizione x,y o la dimensione dei widget mentre ti guiderò attraverso la creazione dell'interfaccia utente. Invece, vi rimifero all'immagine qui sotto e vi lasserò come meglio credete. Tuttavia, vi darò gli attributi importanti che corrisponderanno al codice. Una volta che avete l'applicazione di base di lavoro, si prega di sentirsi liberi di cambiarlo per soddisfare le vostre esigenze. Ecco uno screenshot dell'app GUI appena uscita da Page.

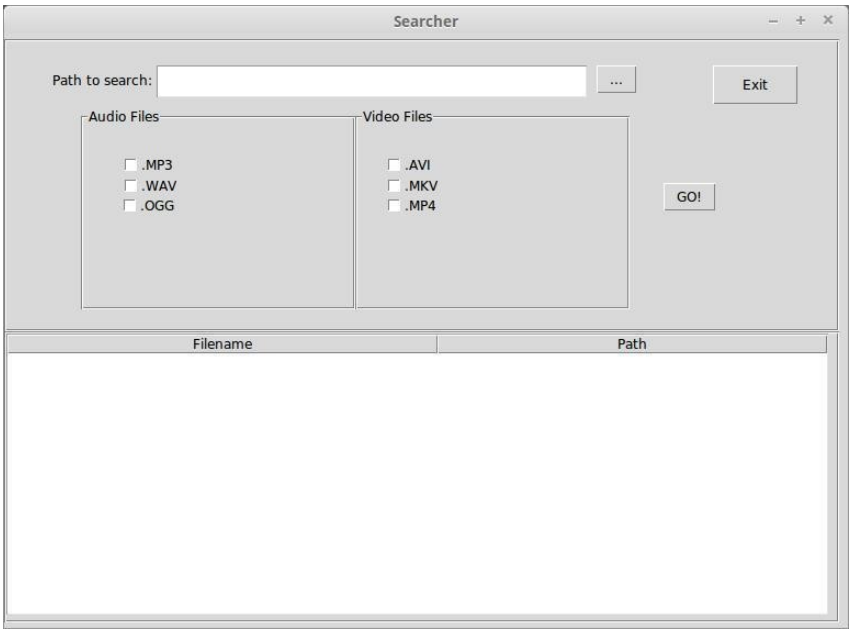

Immagine 12: Interfaccia utente di progettazione del searcher completata

Compilazione dell'interfaccia utente

Inizieremo aprendo pagina e sposteremo la nuova finestra TopMost in un punto vicino al centro dello schermo sia orizzontalmente che verticalmente. Si desidera inoltre impostare il titolo e l'alias su "Searcher". Avremo bisogno di due cornici standard delle stesse dimensioni, che prendono da qualche parte circa la metà della finestra più in alto verticalmente. Impostare l'alias per quello superiore **come frameTop** e quello inferiore su **frameTreeview**. Si desidera salvare questo in una nuova cartella denominata Searcher e il file di progetto .tcl deve essere denominato Searcher.tcl pure.

Ora vicino alla parte superiore della cornice superiore, inserire un widget etichetta, widget di ingresso e due widget pulsante e distanziarli come si vede nell'immagine sopra. Il widget etichetta è semplice da fare, dal momento che si tratta di un'etichetta di testo statica. Impostare l'attributo di testo **su "Percorso per la ricerca:"**. Puoi lasciare l'alias impostato sul valore predefinito "Label1" se lo desideri, o modificarlo in base a ciò che ti si addice. Questo è tutto per il widget etichetta.

Poi ci occuperemo del nostro widget di ingresso. Rendere abbastanza lungo e un po 'più alto rispetto al default. Impostare gli attributi come segue:

alias: **txtPath**
textvariable: **FilePath**

Continuando a muoversi lungo i widget a destra, ci occuperemo del primo dei due widget di pulsanti. Questo dovrebbe essere posizionato vicino all'estremità destra del widget di ingresso e vicino al quadrato. Il testo per il widget sarà solo tre punti ('...'), il che significa che se l'utente fa clic su questo pulsante verrà visualizzata una finestra di dialogo di qualche tipo. In questo caso, verrà visualizzata una finestra che chiede all'utente di selezionare una directory da cercare. Il percorso del file restituito verrà immesso nel widget di immissione a livello di codice dal nostro codice.

alias: **btnSearchPath**
comando: **OnBtnSearchPath**
testo: "**...**”

Il nostro ultimo widget pulsante in questo set permetterà all'utente di uscire dal programma.

alias: **btnExit**
testo: "**Esci**"

Prima di passare, impostare un'associazione per **btnExit** a una funzione denominata **OnBtnExit(e) sul** pulsante del mouse 1.

Si noti che sul pulsante precedente (**btnSearchPath**) è stato utilizzato l'attributo command per impostare la funzione, non un'associazione.

Successivamente abbiamo bisogno di due widget LabelFrame per tenere i nostri pulsanti di controllo in gruppi logici. Impostare la loro dimensione combinata a circa 3/4 dello spazio orizzontale di frameTop. Renderli entrambi circa della stessa dimensione e lungo la stessa posizione verticale (y) in modo che si allineano. Impostare l'etichetta a sinistra su "File audio" e quella destra su "File video". Aggiungere un pulsante che inizierà il processo di ricerca a destra di questi LabelFrames. Torneremo in pochi istanti per impostare gli attributi per questo pulsante.

Ora avremo bisogno di inserire tre caselle di controllo in ogni LabelFrame. Quando ho progettato l'interfaccia utente, li ho messi vicino al lato sinistro del LabelFrame e allineati lungo la sinistra, per consentire aggiunte in futuro, se necessario.

SUGGERIMENTO: *i pulsanti di controllo, per impostazione predefinita, sono impostati per centrare il testo e la casella di controllo entro le dimensioni del widget. Ciò può rendere difficile l'allineamento delle caselle di controllo lungo una linea verticale. Lo modificheremo utilizzando l'attributo di ancoraggio quando abbiamo impostato le definizioni per ogni widget.*

A partire dal file audio Labelframe, mettiamo tre widget di pulsante di controllo al suo interno. Quando ho fatto il design, li ho distanziati con 30 pixel tra di loro verticalmente, ma è possibile disaporre come si desidera. I tre tipi di file che stiamo cercando quando si tratta di file audio sono .mp3, .wav e .ogg. Questo dovrebbe prendersi cura della maggior parte dei file audio che si potrebbe avere sul computer. Anche in questo caso, è possibile aggiungere più in un secondo momento, se si desidera. Partendo dall'alto e scendendo, gli attributi sono:

alias: **chkMP3**
ancoraggio: **W**
giustificare: **SINISTRA**
testo: '**. MP3'**
variabile: **VchkMP3**

Si noti che ho modificato il nome della variabile predefinita in "**Vchk**" più l'estensione del file. Il '**V'** sta per **Variable** e **chk sta** per **CheckButton**. Questo renderà i nostri nomi delle variabili molto più facili da ricordare e avrà più senso quando iniziamo a lavorare con il nostro codice.

alias: **chkWAV**
ancoraggio: **W**
giustificare: **SINISTRA**
testo: **'. WAV'**

variabile: **VchkWAV**

alias: **chkOGG**
ancoraggio: **W**
giustificare: **SINISTRA**
testo: **'. OGG'**
variabile: **VchkOGG**

Ora, nel File video LabelFrame, ripeteremo il processo di posizionamento di tre pulsanti di controllo all'interno del fotogramma. Essi permetteranno all'utente di selezionare tra le seguenti estensioni per i file video comuni, che sono .avi, .mp4 e .mkv. Ancora una volta, ho deciso di posizionarli vicino al lato sinistro del LabelFrame e impilati verticalmente con 30 pixel di spaziatura tra di loro per consentire aggiunte future.

alias: **chkAVI**
ancoraggio: **W**
giustificare: **SINISTRA**
testo: **'. AVI'**
variabile: **VchkAVI**

alias: **chkMP4**
ancoraggio: **W**
giustificare: **SINISTRA**
testo: **'. MP4'**
variabile: **VchkMP4**

alias: **chkMKV**
ancoraggio: **W**
giustificare: **SINISTRA**
testo: **'. MKV'**
variabile: **VchkMKV**

Pensando al futuro del programma, si potrebbe anche aggiungere funzionalità per la ricerca di file di testo, file PDF, foto e chissà cos'altro.
L'ultima cosa che dobbiamo fare per il gruppo frameTop è lavorare con il pulsante "Vai". Ecco i suoi attributi:

alias: **btnGo**
testo: **"GO!"**

Infine associare il pulsante del mouse 1 al pulsante e puntarlo a una funzione denominata **OnBtnGo(e)**.

Ora lavoreremo su frameTreeview. Questa parte è abbastanza facile. Vogliamo inserire un widget ScrolledTreeview in frameTreeview. Il widget ScrolledTreeview si trova nella parte inferiore della casella degli strumenti. Una volta che è in, spostarlo nell'angolo in alto a sinistra del telaio, quindi farlo coprire quasi l'intero fotogramma. Di solito mi piace lasciare uno spazio da 2 a 4 pixel del fotogramma che mostra di agire come un bordo. È possibile lasciare l'alias al valore predefinito "Scrolledtreeview1". L'ultima cosa che dovete fare qui è quello di impostare un'associazione per il pulsante del mouse 1 a OnTreeviewClick(e). Ciò consentirà all'utente di selezionare uno dei file nell'elenco.
Stiamo usando una visualizzazione Albero scorrevole qui per creare un elenco a più colonne che contiene i nomi dei file e i percorsi dei file trovati.

Il Codice

Come sempre, è necessario modificare il codice nel modulo di supporto in modo che corrisponda al codice presentato qui.

Ricorda, se stai utilizzando una versione di Page precedente alla 4.11, che devi inserire **la codifica '-:Remember,**if you are using a version of Page prior to 4.11, that you need to put the '' **- coding:**

utf8 ' come seconda riga nel modulo di supporto come abbiamo discusso l'ultimo capitolo.

Abbiamo bisogno di apportare molte modifiche la sezione importazioni del codice che Page generato per noi per supportare alcune delle cose 'extra' che vogliamo fare. Nel modulo di supporto, è necessario includere i **moduli di libreria di piattaforma, os e os.path** oltre al **modulo sys.** module.

```
importare sys
piattaforma di importazione
importare il
da os.path import join,getsize,exists
```

Purtroppo Python3 non gestisce Tkinter allo stesso modo di Python2, quindi dobbiamo importare le cose in modo un po 'diverso per supportare entrambe le versioni di Python.

Provare:

```
    da Tkinter import
    importare tkMessageBox
    importare tkFont

ad eccezione di ImportError:
    dall'importazione di tkinter
    da tkinter importa messagebox
    da tkinter importa font
    da tkinter import filedialog
```

Un'altra modifica nella versione 4.11 della pagina è nella sezione di codice seguente. La **variabile py3** verrà impostata su True se la versione in cui viene eseguito il codice è Python 3.x e False se è la versione 2.x. Ci sarà

è necessario controllare questo più avanti nel nostro codice per assicurarsi che stiamo usando il codice corretto tra le due diverse versioni di Python.

Pagina 4.10 e versioni precedenti fornisce questo codice per noi:
Provare:

```
    importare ttk

ad eccezione di ImportError:
    importare tkinter.ttk come ttk
    py3 - 1
```

E la pagina 4.11 è riportato in questo codice per noi:
Provare:

```
    importare ttk

ad eccezione di ImportError:
    importare tkinter.ttk come ttk
    py3 - Vero
```

La sezione successiva del codice è la funzione set_Tk_var(). Non ci sono modifiche qui dal codice generato dalla Pagina, ma dovresti vedere come è impostato per noi. Questa sezione imposta le variabili globali definite al momento della creazione dell'interfaccia utente.

```
def set_Tk_var():
    FilePath globale
    FilePath - StringVar()
    VchkMP3 globale
    VchkMP3 - StringVar()
    VchkWAV globale
    VchkWAV - StringVar()
    VchkOGG globale
    VchkOGG - StringVar()
    VchkAVI globale
    VchkAVI - StringVar()
    VchkMP4 globale
    VchkMP4 - StringVar()
    VchkMKV globale
    VchkMKV - StringVar()
```

Ora, inizieremo a definire il codice che viene eseguito quando si fa clic su uno dei pulsanti o altri widget. In primo luogo, ci occuperemo di uno molto semplice... pulsante Esci. Tenere presente che è stato associato il pulsante del mouse-1 a una funzione **denominata OnBtnExit**. Le prime due righe all'interno della funzione vengono create per noi da Page. Per uscire dal programma, usiamo (come abbiamo fatto negli esempi precedenti) la chiamata **destroy_window().**

```
def OnBtnExit(p1):
    print('Searcher_support. OnBtnExit')
    sys.stdout.flush()
    destroy_window()
```

La **funzione OnBtnGo()** è dove si verifica tutto il lavoro pesante. Una volta che abbiamo impostato la cartella di avvio e le estensioni dei file che vogliamo cercare, chiamiamo tutte le funzioni di supporto qui. Naturalmente, avremmo potuto codificarlo per essere una funzione massiccia, ma è molto più leggibile in questo formato. Innanzitutto, deselezionare ScrolledTreeview quindi modificare il cursore del mouse su "occupato". Creare un elenco contenente le estensioni di file che si desidera cercare. Estrarre il percorso di ricerca dei file dalla variabile globale e convertire le estensioni in una tupla. Cancellare l'elenco contenente i risultati della ricerca restituiti in precedenza (se presenti) e chiamare la funzione di ricerca ricorsiva. Infine, caricare i risultati della ricerca nella griglia e riportare il cursore del mouse allo stato predefinito.

```
def OnBtnGo(p1):
    print('Searcher_support. OnBtnGo')
    sys.stdout.flush()
    ClearDataGrid()
    busyStart()
    Funzione BuildExts()
    fp - FilePath.get()
    e1 - tupla (ets)
    #Clear'elenco nel caso in cui l'utente voglia "andare" di nuovo
    del FileList[:] - in python 3.3, è possibile utilizzare list.clear()
    Walkit(fp;e1)
    LoadDataGrid()
    busyEnd()
```

Quando l'utente fa clic sul pulsante 'get search path' (quello con i tre puntini), aprire una finestra di dialogo Tkinter askdirectory per ottenere la directory iniziale per la ricerca. Anziché creare una finestra di dialogo specifica, viene utilizzata una delle tre finestre di dialogo dei file compilate disponibili nel modulo tkFileDialog. I tre includono:

.askopenfile – richiede la selezione di un file esistente
.asksaveasfilename – richiede il nome del file e la directory per salvare o sostituire un file
.askdirectory – richiede un nome di directory.

Mentre noi o l'utente potrebbe semplicemente inserire la directory di partenza nel widget di immissione, questo rende facile per l'utente di inserire il percorso correttamente. Una volta chiusa la finestra di dialogo dopo aver selezionato la directory iniziale, le informazioni sul percorso vengono immesse nel widget di immissione tramite il metodo .set(path) della variabile FilePath. Dobbiamo controllare la variabile **py3** per vedere se è impostata su 1 o 0 (o True o False quando si utilizza la pagina 4.11)

```
def OnBtnSearchPath():
    print('Searcher_support. OnBtnSearchPath')
    sys.stdout.flush()
    if (py3 - 1) o (py3 - True):
        percorso : filedialog.askdirectory()
    Altro:
        percorso : tkFileDialog.askdirectory()
    FilePath.set(percorso)
```

La **funzione OnTreeviewClick()** è impostata per un utilizzo successivo.

```
def OnTreeviewClick(e):
    print('Searcher_Support.OnTreeviewClick')
    sys.stdout.flush()
    "Lo useremo nel prossimo capitolo.
```

Quando iniziamo il programma, vogliamo che tutti i widget del pulsante di controllo siano deselezionati. Usiamo la funzione **BlankChecks()** per impostare le variabili associate a ciascuno dei pulsanti di controllo su uno stato 0 o unchecked.

```
def BlankChecks():
    VchkAVI.set('0')
    VchkMKV.set('0')
    VchkMP3.set('0')
    VchkMP4.set('0')
    VchkOGG.set('0')
    VchkWAV.set('0')
```

La **funzione BuildExts()** crea un elenco di estensioni che verranno utilizzate dalla funzione ricorsiva filename (Walkit()). Controlliamo semplicemente la variabile associata a ciascuno dei nostri widget Checkbox per vedere se ha un valore di "1". In tal caso, aggiungiamo il testo dell'estensione all'elenco chiamato "exts". Quando entriamo nella funzione, vogliamo svuotare l'elenco, nel caso in cui l'utente cambia la scelta di un'estensione e passa di nuovo attraverso il processo. Lo facciamo usando il comando 'del list[:]'. Python3 consente un metodo list.clear() che potrebbe sostituire questo.

```
def BuildExts():
    del exts[:]          - Cancellare l'elenco delle estensioni, quindi ricompilarlo...
                    se VchkAVI.get() è '1':
                      exts.append(".avi")
                    se VchkMKV.get() è '1':
                     exts.append(".mkv")
                    se VchkMP3.get() è '1':
                     exts.append(".mp3")
                    se VchkMP4.get() è '1':
                     exts.append(".mp4")
                    se VchkOGG.get() è '1':
                      exts.append(".ogg")
                    se VchkWAV.get() è '1':
                     exts.append(".wav")
```

La **funzione busyStart()** cambia il cursore del mouse in un cursore 'watch' per la finestra principale e tutti i widget figlio, ad eccezione del widget di immissione. Questo funziona bene sotto Linux, ma il sistema operativo Windows ha un problema di eseguire questa operazione mentre viene eseguita la scansione ricorsiva del nome file.

```
def busyStart(newcursor-None):
    preBusyCursors globale
    print('busyStart')
    se non newcursor:
        newcursor - occupatoCursor
    newPreBusyCursors
```

```
    forcomponent in busyWidgets: newPreBusyCursors[component] -
        component['cursor'] component.configure(cursor-newcursor)
        component.update_idletasks()

    preBusyCursors (newPreBusyCursors, preBusyCursors)
```

La **funzione busyEnd()** riporta il cursore del mouse al cursore predefinito.

```
def busyEnd():
    preBusyCursors globale
    print('busyEnd')
    se non preBusyCursors:
        Ritorno
    oldPreBusyCursors - preBusyCursors[0]
    preBusyCursors - preBusyCursors[1]
    per il componente in busyWidgets:
        Provare:

            component.configure(cursore-oldPreBusyCursors[componente])

            Passare
        component.update_idletasks()
```

Ecco il vero cuore del nostro programma. La **funzione Walkit()** accetta la cartella iniziale e l'elenco delle estensioni, convertite in tupla, e cammina in modo ricorsivo alla ricerca di un file con un'estensione corrispondente a quella che stiamo cercando. Se viene trovato un file che corrisponde, il nome file e il percorso vengono aggiunti a un altro elenco (**fl**) che viene quindiaggiunto all'elenco da restituire. Continua così fino a quando tutti i file in tutte le cartelle sotto la cartella di avvio sono stati controllati.

```
def Walkit(percorso musicale,estensioni):
    rcntr - 0
    fl - []
    per root, dirs, file in os.walk(musicpath):
        rcntr - 1 - Questo è il numero di cartelle che abbiamo camminato per il file in [f per f
        nei file se f.endswith(extensions)]:
            fl.append(file)
            fl.append(radice)
            FileList.append(fl)
            fl-[]
```

La **funzione init()** viene eseguita all'avvio del programma prima che venga visualizzata la finestra principale (widget topmost). Le prime quattro righe (che terminano con 'root e top') vengono generate da Page. Il resto della funzione si occupa del nostro codice di avvio.

```
def init(top, gui, sargs, kwargs):
    globale w, top_level, radice
    w - gui
    top_level - in alto
    radice - superiore
```

Iniziamo dichiarando tre variabili globali, **treeview**, **exts** e **FileList**. **Exts** e **FileList** abbiamo

già discusso. Treeview viene utilizzato per rendere più semplice per noi fare riferimento nel codice al widget ScrolledTreeview. Abbiamo anche impostato **exts** e **FileList** per liste vuote. Successivamente chiamiamo la funzione **BlankChecks()** per cancellare i widget del pulsante di selezione, impostare la vista ad albero variabile **globale** in modo che punti al nostro widget Scrolledtreeview1. Si noti che si utilizza **w.Scrolledtreeview1**. Il '**w.**' si riferisce alla nostra GUI e quando facciamo direttamente chiamate ai nostri widget dobbiamo anteporre il '**w.**'. Infine, abbiamo impostare le informazioni per le nostre funzioni cursore occupato.

```
Il nostro codice inizia qui
visualizzazione ad albero globale, exts, FileList
exts []
Percorso File(StringVar)
FileList:[]
#-------------
Controllo Vuoto()
treeview - w.Scrolledtreeview1
SetupTreeview()
#-------------
globale occupatoCursor,preBusyCursors,busyWidgets
busyCursor - 'guarda'
preBusyCursors - Nessuno
busyWidgets ( radice, )
```

La **funzione SetupTreeview()** imposta il numero di colonne e le intestazioni per ogni colonna dell'elenco globale **ColHeads** utilizzando il metodo .configure. In questo caso, avremo solo due colonne denominate 'Nome file' e 'Percorso'. Controlliamo anche la variabile py3 per vedere se stiamo eseguendo in Python3 o Python2 per effettuare la chiamata corretta al modulo del tipo di carattere.

Si noti che la funzione SetupTreeview deve essere chiamata prima di tentare di caricare i dati nella griglia.

```
def SetupTreeview():
    ColHeads globali
    ColHeads - ['Nomefile','Percorso'] treeview.configure(columns, ColHeads,
    show-"headings") per col in ColHeads:
        treeview.heading(col,text,col.title(),command'lambda c'è col: sortby(treeview,c,0)) - regola la larghezza della colonna
        sulla stringa di intestazione
        if (py3 - 1) o (py3 - True):
            treeview.column(col, width : font. Font().measure(col.title()))
        Altro:
            treeview.column(col, width : tkFont.Font().measure(col.title()))
```

La **funzione ClearDataGrid()** rimuoverà tutti i dati dal widget Scrolledtreeview. Se avete intenzione di utilizzare il Scrolledtreeview in altri progetti, questo sarebbe un utile da tenere nel vostro kit di strumenti. Fondamentalmente, tutto ciò che fa è camminare attraverso il widget treeview ed elimina ciascuno degli elementi di dati uno per uno.

```
def ClearDataGrid():
    #print("Into ClearDataGrid")
    per c in treeview.get_children(''):
        treeview.delete(c)
```

La **funzione LoadDataGrid()** cancella prima il widget treeview e quindi accetta i risultati restituiti dalla funzione **Walkit()** e carica ogni risultato in una riga nel widget Treeview. Il resto del codice si adatterà alla larghezza delle colonne per adattarla al valore più lungo. Anche in questo caso, dobbiamo controllare se stiamo usando Python3 o Python2 per effettuare la chiamata corretta al modulo del tipo di carattere.

```
def LoadDataGrid():
    ColHeads globali
    ClearDataGrid()
    per c in FileList:
        treeview.insert('','end',values
        - regolare la larghezza della colonna se necessario per adattarsi a ogni valore
        per ix, val in enumerate(c):
            if (py3 - 1) o (py3 - True):
                col_w carattere di tipo carattere. Font().measure(val)
            Altro:
                col_w : tkFont.Font().measure(val)
            se treeview.column(ColHeads[ix],width,None)<col_w:
                treeview.column(ColHeads[ix], width col_w)
```

Infine, abbiamo una funzione chiamata **sortby() che,** quando si fa clic su un'intestazione di colonna, ordinerà la visualizzazione Albero. Questo è associato al ScrolledTreeview nella funzione SetupTreeview().

```
def sortby(albero, col, decrescente):
    """ordinare il contenuto della struttura quando si fa clic su un'intestazione di colonna"""
    - afferrare i valori per ordinare
                                data : [(tree.set(child, col), child)
                            per il bambino in tree.get_children('')]
    : se i dati da ordinare sono modifica numerica in float #data ,
    change_numeric (dati)
    : ora ordina i dati sul posto
    data.sort(inverso/decrescente)
    per ix, elemento in enumerate(data):
        tree.move(elemento[1], '', ix)
    : consente di cambiare l'intestazione in modo che si elame nella direzione opposta
    tree.heading(col, command,lambda col-col: sortby(tree, col,
        int(non decrescente)))
```

Quando si esegue il programma, è possibile fare clic su una voce nella visualizzazione Albero, ma non accadrà nulla, ad eccezione di una breve stampa alla finestra del terminale. Mentre questo può essere un programma utile per trovare file multimediali specifici nella libreria, non sarebbe bello se si potesse riprodurre un file selezionato, audio o video, semplicemente eseguendo questo programma e quindi facendo clic su di esso nella griglia di visualizzazione ad albero?

Questo è quello che faremo nel prossimo capitolo.

Capitolo 4 – Collegamento di un lettore multimediale a Searche

In questo capitolo, creeremo un lettore multimediale per file audio e video, che è collegato al programma Searcher che abbiamo appena creato.

Alcune delle cose che imparerai in questo capitolo sono:

Widget barra di
scorrimento Il
metodo '.after'

Barra dei menu dell'applicazione di
programmazione completamente
guidata dagli eventi

Aggiunta di un secondo (o più) form a un'applicazione.

Dovrai scaricare e installare il modulo libreria del lettore VLC (python-vlc) e il programma VLC stesso se non lo hai già. È possibile installare il modulo **da sudo pip installare python-vlc**.

La maggior parte del codice e il layout proviene dalla pagina web VideoLan.org esempi https://wiki.videolan.org/python_bindingse ho apportato alcune modifiche ad esso per semplificare le cose un po '.

Ecco come appare il progetto finito:

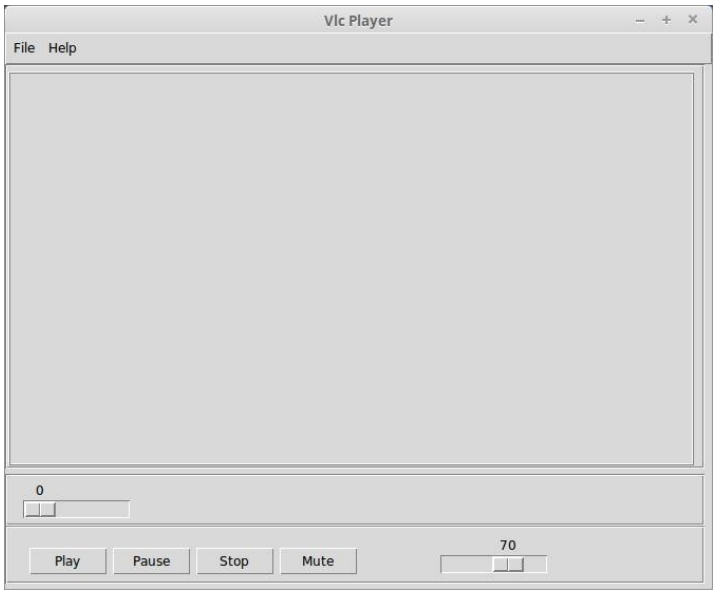

Immagine 13: Lettore VLC

Compilazione dell'interfaccia utente

Carica pagina e ottieni il tuo nuovo widget Toplevel. Impostare l'alias **su vlcplayer** e il titolo su "**VLC Player**" e le dimensioni e la posizioneapprossimate sullo schermo. Ho fatto il mio 670 x 500. Salvare il progetto nella stessa cartella in cui è stato salvato il progetto Searcher dall'ultimo capitolo di **vlcplayer**.

Creazione della barra dei menu

Nella finestra principale della Pagina, selezionare "Widget Modifica barra dei menu..." e verrà visualizzata una nuova finestra che contiene l'Editor menu.

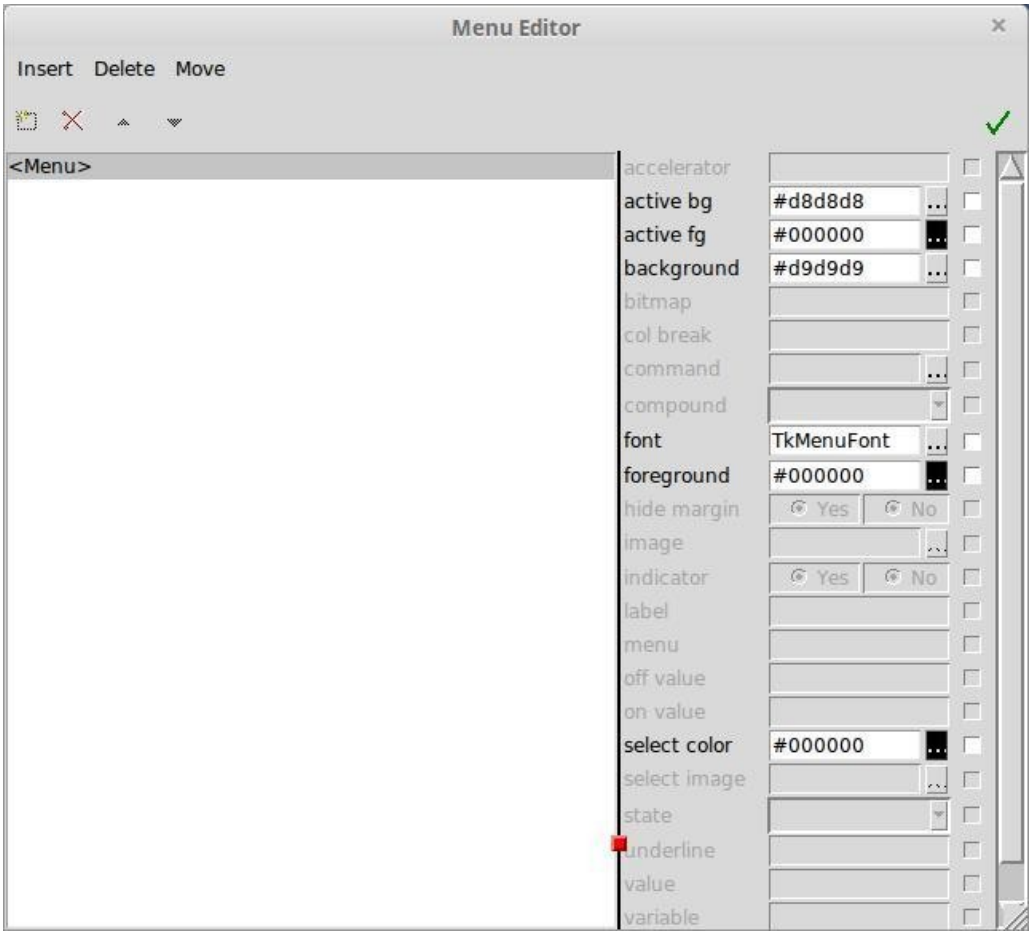

Immagine 14: Editor menu

A questo punto, selezionare 'Inserisci Nuova cascata'. Questo inizierà il processo. Normalmente, la prima voce della barra dei menu è

Operazioni sui file, ad esempio Nuovo, Apri, Salva, Salva con nome, Esci e così via. In questo caso verrà rinominato l'elemento appena creato da 'NewCascade' a 'File'. Fare clic su di esso e nella sezione attributi dell'editor di menu, modificare l'attributo label in 'File'. Ora seleziona 'Inserisci Nuovo comando'. Modificare l'attributo label in 'Open', quindi ripetere questo processo per creare un nuovo comando denominato 'Exit'.
Fare di nuovo clic sull'opzione 'Apri' e impostare l'attributo del comando su 'mnuFileOpen'. (È possibile modificare direttamente il testo o fare clic sul pulsante di modifica per visualizzare una finestra.) Ora fare clic sull'opzione Esci e impostare il suo comando su 'mnuFileExit'.
Quindi fare clic su '<Menu>' e selezionare 'Inserisci Nuova cascata'. Verrà creata una nuova voce di menu "livello superiore" sotto il set di menu File. Rinominarlo da 'NewCascade' a 'Help'. Ora seleziona 'Inserisci Nuovo comando' e rinominarlo in 'Informazioni su'. Impostare l'attributo del comando su 'mnuHelpAbout'.
Fare clic sul segno di spunta verde in alto a destra per chiudere l'Editor menu. Salvare il file di progetto tcl, generare il modulo di codice python e il modulo di supporto.
Se si guarda il modulo di supporto, si vedrà che Page ha generato le tre funzioni di menu (mnuFileOpen(), mnuFileExit() e mnuHelpAbout()) per noi.
Successivamente, abbiamo bisogno di tre widget cornice. Il primo, terrà un widget canvas che fungerà da "schermo" video quando riproduciamo i video. Il secondo terrà un widget cursore per il posizionamento all'interno del file multimediale. Si aggiornerà automaticamente durante la riproduzione del file multimediale. Il terzo, terrà quattro pulsanti e un widget cursore.
Il primo widget cornice dovrebbe essere di circa 3/4 del widget Livello superiore verticalmente e riempire lo spazio orizzontalmente. Denostala 'frameVideo'. Il prossimo dovrebbe prendere la metà dello spazio verticale rimanente e il riempimento dello spazio orizzontalmente pure. Denostala frameSlider. L'ultimo dovrebbe avere circa le stesse dimensioni del secondo e denonirlo frameButtons.

Ora posizionate un widget canvas nel widget frameVideo. È possibile utilizzare il nome alias predefinito (Canvas1).

Si desidera fare in modo che riempia l'intera cornice.

Nella corniceSlider, posizionate un widget cursore orizzontale. Impostare gli attributi come segue:

alias: **TimeSlider**
comando: **ScaleSel**
lunghezza: **500**
a: **1000**
variabile: **ScaleVar**

Successivamente nel widget frameButtons, è necessario posizionare quattro widget di pulsanti e un cursore orizzontale lungo la stessa linea orizzontale. Ogni pulsante deve avere una larghezza di 75 pixel e spaziarsi in modo uniforme. Ecco la

attributi per i quattro pulsanti da sinistra a destra:

alias: **btnPlay**
testo:riprodurre

alias: **btnPause**
testo:pausa

alias: **btnStop**
testo:Interrompere

alias: **btnMute**
testo:disattivare l'audio

E gli attributi per il cursore orizzontale:

alias: **VolumeSlider**
comando: **VolumeSel**
variabile: **VolumeVar**

Se si desidera, è possibile inserire un'etichetta **tra btnMute** e **VolumeSlider** con il testo di "Volume:", ma questo è strettamente facoltativo.

Infine, abbiamo bisogno di legare il pulsante del mouse-1 a ciascuno dei quattro pulsanti per:

btnPlay: **OnBtnPlay**
btnPause: **OnBtnPause**
btnStop: **OnBtnStop**
btnMute: **OnBtnMute**

Ora possiamo inserire il nostro codice.

Il Codice

Il funzionamento di questo programma è duplice. In primo luogo, può essere lanciato come un programma autonomo e utilizzando la barra dei menu, è possibile selezionare un file multimediale da giocare. In secondo luogo, sarà chiamato, attraverso alcune piccole modifiche, dal nostro programma Searcher che passerà il percorso del file e il nome del file al programma del lettore e avviare la riproduzione del file multimediale.

Lamenteremo con il file di supporto. La prima cosa che vogliamo fare è impostare le importazioni, in modo da cambiare ciò che la pagina ci ha dato a:

```
# Importare la libreria VLC
import vlc
# Importare le librerie standard importano
sys
import os import
pathlib import time
import platform
```

Provare:

```
    da Tkinter import
    importazione tkFileDialog

ad eccezione di ImportError:
    dall'importazione di tkinter
    da tkinter importa messagebox
    da tkinter importa font
    da tkinter import filedialog
```

Provare:

```
    importare ttk

ad eccezione di ImportError:
    importare tkinter.ttk come ttk
    py3 - 1
```

Dobbiamo aggiungere alla funzione init che è già stata creata per noi. Aggiungeremo le linee che sono in grassetto sotto la radice della **linea ::**

```
def init(top, gui, sargs, kwargs):

    globale w, top_level, radice
    w - gui
    top_level - in alto
    radice - superiore

    istanza globale, player
    globale timeslider_last_val, timeslider_last_update
    timeslider_last_val 0
    timeslider_last_update 0
    Istanza: vlc. Instance()
    player - Instance.media_player_new()
    se len(args) !
        temp - list(args)[0]
        dirname : temp[0]
        nomefile : temp[1]
        Media: Instance.media_new(str(os.path.join(dirname, filename)))
        player.set_media (Media)
        se platform.system() è 'Windows':
            player.set_hwnd(self. GetHandle())
        Altro:
            player.set_xwindow(GetHandle())
        OnBtnPlay()
    Altro:
```



```
        OnOpen()
    VolumeVar.set(70)
```

Stiamo verificando la lunghezza della variabile args che viene passata nella funzione init. Stiamo facendo questo per consentire al programma di essere eseguito sia in un metodo autonomo o chiamato dal progetto di ricerca che abbiamo fatto nell'ultimo capitolo. Quando il lettore multimediale viene chiamato da searcher, passerà il nome della directory e il nome del file multimediale da riprodotto attraverso la variabile . Abbiamo anche bisogno di fare gran parte del lavoro che viene fatto nella **funzione OnOpen(),** quindi abbiamo impostato l'handle per il widget canvas prima di chiamare la funzione **OnBtnPlay().**
Successivamente inizieremo a lavorare sulle nostre funzioni di pulsante. Si noti che usiamo il **metodo .config()** per modificare l'attributo di testo del pulsante mute.

```
def OnBtnMute(p1):
    sys.stdout.flush()
    is_mute - player.audio_get_mute()
    print("is_mute - {0}".format(is_mute))
    se is_mute n. 1:
        w.btnMute.config(testo : 'Mute')
    Altro:
        w.btnMute.config(testo : 'Riattiva audio')
    player.audio_set_mute (non is_mute)
    VolumeVar.set(player.audio_get_volume())
```

Qui chiamiamo semplicemente la **funzione player.pause()** per mettere in pausa e annullare la stampa del lettore.

```
def OnBtnPause(p1):
    sys.stdout.flush()
    player.pause()
```

La **funzione OnBtnPlay()** chiamerà la funzione **OnOpen()** per far funzionare le cose. Usiamo anche il .after piuttosto che un ciclo semplice o un timer threadd separato. Tkinter utilizza il **metodo root.after** per creare un timer per noi, tuttavia non è abbastanza preciso come un timer separato, ma è abbastanza preciso per gestire le cose di cui abbiamo bisogno. Otteniamo il Timer_id quando chiamiamo il metodo **root.after**iniziale (time,function) per impostare le cose. Poiché è la prima volta che lo chiamiamo, usiamo 0 come tempo e impostamo la funzione da chiamare alla nostra funzione **OnTick().** Una volta che si entra nella funzione **OnTick(),** l'ultima riga imposta il tempo per la chiamata successiva, che nel nostro caso è di circa 1000 millisecondi.

```
def OnBtnPlay(p1-Nessuno):
    sys.stdout.flush()
    globale Timer_id

    se non player.get_media():
        OnOpen()
    Altro:
        Provare ad avviare il supporto, se l'operazione non riesce visualizza un messaggio di errore

        se player.play() -1:
```



```
            errorDialog("Impossibile riprodurre.")
    Timer_id : root.after(0,OnTick())
```

Quando clicamo sul pulsante di arresto, chiamiamo la funzione **player.stop()** e si imposta la posizione del cursore temporale su 0.

```
def OnBtnStop(p1):
    sys.stdout.flush()
    player.stop()
    : consente di reimpostare il dispositivo di scorrimento del tempo
    w.TimeSlider.set(0)
```

Il file Il comando di menu **exit destroy_window()** come abbiamo fatto molte volte in precedenza.

```
def OnMnuFileExit():
    sys.stdout.flush()
    destroy_window()
```

Dalla barra dei file Aprire il comando di menu, chiamiamo **la funzione OnOpen()** per far funzionare le cose quando si è in modalità autonoma.

```
def OnMnuFileOpen():
    sys.stdout.flush()
    OnOpen()
```

Concretizzaremo l'Aiuto A proposito di comando di menu più tardi, ma ho messo qui semplicemente per completezza.

```
def OnMnuHelpAbout():
    sys.stdout.flush()
```

La **funzione ScaleSel()** consente all'utente di impostare la posizione lorda nel file multimediale in modo che analogico in avanti/riavvolgimento. L'utente deve fare clic e trascinare il cursore "pulsante" invece di fare clic all'interno
il vassoio per farlo "saltare" alla posizione del mouse, tuttavia se l'utente tiene il pulsante del mouse-1 verso il basso
all'interno del vassoio, il cursore salterà rapidamente nei passaggi. Assegniamo la **variabile nval** al **ScaleVar**
variabile utilizzando il **metodo .get(),** quindi confronta il valore corrente con l'ultimo valore

(**timeslider_last_val**) per ridurre al minimo la possibilità di un ciclo infinito. Se sono diversi, aggiorniamo il

l'ultima volta che il cursore è stato aggiornato, convertire la posizione in millisecondi e aggiornare la posizione del giocatore

con la **set_time()** .

```
def ScaleSel(-args):
    global timeslider_last_update, timeslider_last_val sys.stdout.flush()
    se il lettore è Nessuno:
        Ritorno
    nval - ScaleVar.get()
    muscolo : str(nval)
    se timeslider_last_val !
```



```
        timeslider_last_update : time.time()
        mval - "%.0f" % (nval : 1000)
        player.set_time(int(mval)) - prevede millisecondi
```

La **funzione VolumeSel()** (come suggerisce il nome) consente all'utente di modificare il volume del file riprodotto. Questa funzione viene chiamata ogni volta che viene spostato il dispositivo di scorrimento del volume. Se il lettore non è stato avviato, la funzione viene semplicemente chiusa. In caso contrario, otteniamo il valore intero del cursore (tramite la variabile VolumeVar) e lo assegniamo a una variabile chiamata volume per semplicità. Quindi "normalizziamo" il valore e chiamiamo la funzione audio_set_volume() **del** giocatore.

```
def VolumeSel(-args):
    se il lettore è Nessuno:
        Ritorno
    volume : int(VolumeVar.get())
    se volume > 100:
        volume : 100
    volume elif : 0:
        volume : 70
    resp - player.audio_set_volume (volume)
```

La **funzione OnOpen()** imposta il nome file e il percorso del file multimediale da giocare utilizzando una finestra di dialogo del file di stile **askopenfilename** fornita da Tkinter.

```
def OnOpen():
    p - pathlib. Percorso(os.path.expanduser("""))
    if (py3 - 1) o (py3 - True):
        fullname : filedialog.askopenfilename(initialdir , p, title , "scegli il file", filetypes (("tutti i file", "."mp4
files",".mp4")))
    Altro:
        fullname : tkFileDialog.askopenfilename(initialdir , p, title , "scegli il file", filetypes (("tutti i file", "."),("mp4
files",".mp4")))
    se os.path.isfile(fullname):
        dirname os.path.dirname(nome completo)
        nomefile : os.path.basename(nome completo)
        print("{0} - {1}".format(dirname,filename))
        Media: Instance.media_new(str(os.path.join(dirname, filename)))
        player.set_media (Media)
        se platform.system() è 'Windows':
            player.set_hwnd(self. GetHandle())
        Altro:
            player.set_xwindow(GetHandle())
        OnBtnPlay()
```

Ecco la nostra funzione di callback "timer" **OnTick()**. Usiamo questo per fare qualche manutenzione alle varie variabili e widget. Poiché la lunghezza "pubblicata" di un file multimediale può cambiare durante la riproduzione, aggiorniamo la lunghezza del cursore impostando l'attributo "to" sulla lunghezza del file raccolto da una chiamata al metodo **get_length() del lettore** e aggiorniamo la posizione corrente sul dispositivo di scorrimento temporale.

```
def OnTick():
    global timeslider_last_update, timeslider_last_val, Timer_id se il giocatore è Nessuno:
        Ritorno
    Dal momento che il self.player.get_length può cambiare durante la riproduzione,
    : ri-impostare il timeslider sull'intervallo corretto.
    length : player.get_length()
    dbl - lunghezza : 0,001
    w.TimeSlider.config(to'dbl)
    : consente di aggiornare l'ora sul dispositivo di scorrimento
    tyme : player.get_time()
    se tyme -1:
        tyme : 0
    dbl - tyme - 0,001
    timeslider_last_val : ("%.0f" % dbl) - ".0"
    Non voglio cambiare a livello di programma il cursore mentre l'utente sta scherzando con esso.
    Attendere 2 secondi dopo che l'utente lascia andare il dispositivo
    di scorrimento se time.time() > (timeslider_last_update 2.0):
        w.TimeSlider.set(dbl)
    Timer_id : root.after(1000,OnTick)
```

La **funzione GetHandle()** ci permette di ottenere l'ID widget del widget canvas, che viene utilizzato dalla funzione **OnOpen()** in modo che il lettore sappia dove eseguire il rendering del video del file multimediale.

```
def GetHandle():
    w.Canvas1.winfo_id()
```

Infine, abbiamo la **funzione errorDialog()** che accetta qualsiasi messaggio che vogliamo essere visualizzato all'utente come una finestra di messaggio in stile **showerror messagebox**.

```
def errorDialog(messaggio di errore):
    tkMessageBox.showerror('Errore',messaggio di errore)
```

La casella Informazioni su

Mentre Tkinter ha un certo numero di finestre di dialogo create per noi già, stranamente non hanno un about box. Così, ho creato uno generico che presenterò qui. Di seguito è riportata una schermata della casella Informazioni completa utilizzata nel programma Lettore video. Potete vedere, ci sono quattro widget di etichette che sono impostati con un attributo di rilievo affondato uno sopra l'altro, un widget ScrolledText e un widget di pulsante per chiudere la casella Dita quando vogliamo che vada via.

Il modulo Informazioni sulla scatola

Quando create la casella Informazioni su, assicuratevi di salvare il codice nella stessa cartella utilizzata per il progetto del lettore multimediale, che dovrebbe essere la stessa del progetto di ricerca.

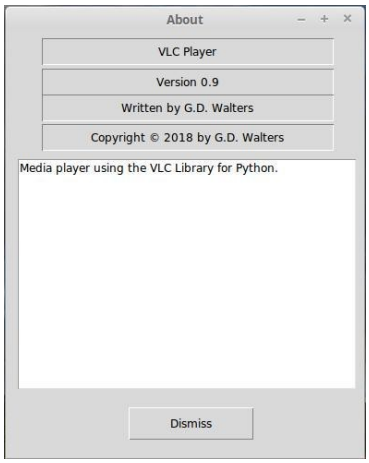

Immagine 15: La casella Generic About

Rendere il widget TopMost un po ' rettangolare e posizionato al centro dello schermo. Impostare l'attributo Titolo **su "Informazioni su"** e l'alias **su "Informazioni su".**

Ora posizionate quattro widget etichetta nel widget TopMost, vicino a uno sopra l'altro, ugualmente distanziati verticalmente e tutti con la stessa larghezza. Impostare **l'attributo di** rilievo su "**sunken**" per tutti e quattro. Non ho modificato il testo per nessuno di essi, dal momento che ognuno avrà i loro attributi di testo modificati in fase di esecuzione. Denoiteli, dall'alto verso il basso come:

lblProgName, lblVersion, lblAuthor e **lblCopyright**.

Successivamente, posizionate un widget ScrolledText che occupa la parte del leone del widget TopMost (lasciando spazio a un pulsante abbastanza grande vicino alla parte inferiore). L'unico attributo che deve essere modificato qui è l'attributo "wrap", che deve essere impostato su "WORD". Tuttavia, è necessario fare clic sul "Testo" sotto la voce Di scorrimento nell'albero Widget per essere in grado di arrivarvi. Mentre buona pratica di programmazione suggerisce che abbiamo impostato l'alias del widget su qualcosa oltre al valore predefinito, non mi sono preoccupato.

Infine, abbiamo bisogno di un widget pulsante standard centrato orizzontalmente e verticalmente nello spazio rimanente e un po 'grande. Questo sarà il nostro pulsante Ignora. Impostare il relativo alias **su btnDismiss**, impostare l'attributo di testosu "**Dismiss**" e associare un eventobutton-1 del mouse in modo che punti alla funzione **OnBtnDismiss**. Salvare il progetto ed entrambi i moduli di codice con il nome "**GenAboutBox**".

Informazioni sul codice scatola

Il codice per il About Box è davvero molto semplice con un paio di elementi importanti.

È molto importante aggiungere la riga seguente all'inizio del file di supporto se non è già presente.

```
Codifica - - -
```

Se non lo fai, si otterrà il temuto "**SyntaxError: Carattere non ASCII ''xe2' nel file...** " errore quando si tenta di avviare la casella generica su, poiché si utilizza un carattere Unicode per supportare il simbolo "©".

Ecco le importazioni standard.

```
importare sys
Provare:
    da Tkinter import

    dall'importazione di tkinter
Provare:
    importare ttk

ad eccezione di ImportError:
    importare tkinter.ttk come ttk
    py3 - 1
```

Nella funzione **OnBtnDismiss** viene utilizzato **root.withdraw()** per *nascondere* la casella about anziché **destroy_window()** che normalmente usiamo per chiudere il programma. Se dovemo usare il **destroy_window()** qui, chiuderebbe l'intero programma.

```
def OnBtnDismiss(p1):
    print('GenAboutBox_support. OnBtnDismiss')
    sys.stdout.flush()
    root.withdraw()
```

Nella funzione **LoadForm(),** caricheremo (come suggerisce il nome della funzione) tutte le informazioni inviate dal nostro lettore video nelle etichette e nel widget ScrolledText. Per le etichette, usiamo il **.configure(attributo -)** per impostare il testo dell'etichetta. Per il widget Testo scorrevole, viene utilizzato il metodo . **insert()** per inserire il testo nella casella di testo alla fine di tutto ciò che è già lì, che in questo caso non è nulla.

```
 def LoadForm():

    temp globale
copyright_symbol s"u00A9"
w.lblProgName.configure(testo : temp[0])
w.lblAuthor.configure(testo : "Scritto da {0}".format(temp[1]))
w.lblVersion.configure(testo : "{0}".format(temp[2]))
w.lblCopyright.configure(testo : temp[3])
w.Scrolledtext1.insert(FINE,temp[4])
```

La funzione init è praticamente la stessa di sempre, tranne che stiamo aggiungendo due righe. Uno per ottenere il

dati inviati dal programma multimediale che include 5 elementi (Nome programma, Nome autore, Versione del programma, Avviso di copyright e informazioni sul programma) nonché la chiamata a LoadForm.

```
def init(top, gui, sargs, kwargs):
    globale w, top_level, radice, temp
    w - gui
    top_level - in alto
    radice - superiore
    temp - list(args)[0]
    LoadForm()
```

Le **destroy_window** e le **funzioni if __name__** sono lo standard che sono già state create per noi da Page senza alcuna modifica, ma le ho incluse solo per essere complete.

```
def destroy_window():
    Funzione che chiude la finestra.
    globale top_level
    top_level.destroy()
    top_level - Nessuno

se __name__ '__main__':
    importare GenAboutBox
    GenAboutBox.vp_start_gui()
```

Collegamento della casella Informazioni su al lettore multimediale

Ora che tutto il duro lavoro è stato fatto, abbiamo bisogno di collegare i file di programma About Box al nostro Media Player. Questo è molto semplice e richiede solo alcune righe di codice da aggiungere al file di supporto di Media Player.

In primo luogo, abbiamo bisogno di aggiungere una riga da qualche parte che importa il nostro progetto About Box come una libreria. Mi piace metterlo in cima al file di supporto con il resto delle importazioni, solo per tenerli tutti insieme. Quindi, sotto la riga che dice **piattaforma di importazione**,inserire:

```
importare GenAboutBox
```

Ora nella funzione, abbiamo bisogno di aggiungere le righe che definiscono le variabili che passeremo al programma About Box. Ecco la funzione completa, incluse le righe che Page ha creato per noi.

```
def OnMnuHelpAbout():
    print('vlcplayer_support. OnMnuHelpAbout')
    sys.stdout.flush()
    Il nostro codice da qui in basso
    copyright_symbol s"u00A9"
    ProgName - "VLC Player"
    Autore : "G.D. Walters"
    Versione : '0.9'
```

```
Copyright - "Copyright" copyright_symbol " 2018 di G.D. Walters" Info - "Lettore multimediale che
utilizza la libreria VLC per Python." GenAboutBox.create_About(root;
[ProgName,Author,Version,Copyright,Info])
```

Suddividendo, definiamo**il simbolo "©"** per copyright_symbol variabile come carattere Unicode. Successivamente definiamo il resto delle variabili (ProgName, Author, Version, Copyright e Info) assegnando loro stringhe per prepararle a passare alla funzione create_About del programma About Box.

Infine chiamiamo la funzione **create_About** (il punto di ingresso all'interno del nostro prossimo programma da mostrare) con le variabili appropriate, i nostri dati vengono passati come un elenco. Ora, diamo un'occhiata alla funzione create_About nel file GenAboutBox.py, che è stato creato per noi da Page.

```
def create_About(root, args, kwargs):
    '''Punto di partenza quando il modulo viene importato da un altro programma.'' globale w,
    w_win, rt
    rt - radice
    w - Livello superiore (radice)
    top - Informazioni su (w)
    GenAboutBox_support.init(w, top, args, kwargs)
    ritorno (w, superiore)
```

Questa funzione esiste in qualsiasi file python principale che Page crea e viene utilizzato per permetterci di avere più programmi di forma. Quando questa funzione viene chiamata, chiama a sua volta **la funzione .init()** che si trova nel file di supporto, che viene caricata prima che il programma mostri la finestra principale del form aggiuntivo, in questo caso la casella Informazioni.
Si noti che la **chiamata passa gli argomenti** alla **funzione .init().** Questo è l'elenco delle variabili che verranno passate per riempire i widget etichetta e il widget Testo scorrevole. Quando **.init() ottiene** queste variabili, è possibile eseguire il cast in un elenco a cui è possibile accedere come nella funzione LoadForm all'interno del modulo di supporto della casella Informazioni su.
Ciò consente al programma di avere più moduli e passare informazioni tra di essi.

Collegamento di Media Player al programma Searcher

Ora dobbiamo modificare il programma Searcher, più o meno come abbiamo fatto quando abbiamo aggiunto la casella Informazioni sul lettore multimediale.

Nel modulo di supporto per Searcher, è necessario importare il programma di lettore multimediale,

```
importare vlcplayer
```

in alto con le altre importazioni. Aggiungere quindi la riga seguente alla **funzione OnTreeviewClick(e).**

```
riga treeview.identify_row(e.y)
col = treeview.identify_column(e.x)
```

nomefile : treeview.set(riga,0)
path : treeview.set(riga,1)
vlcplayer.create_VLC_Player(radice;[percorso;nomefile])

Qui, otteniamo la riga e la colonna in cui si è verificato il clic del pulsante del mouse-1 in poi usiamo queste informazioni per impostare le variabili di percorso e nome file dalle informazioni nella griglia del widget ScrolledTreeView. Infine chiamiamo la funzione create_VLC_Player con le due variabili.

Questo è tutto. Salvare tutti i file e assicurarsi che facendo clic su tutte le voci di menu e pulsanti che tutto funziona come previsto.