# Python Programming

Módulo 06



EDUCACIÓN IT

# Aplicaciones de escritorio

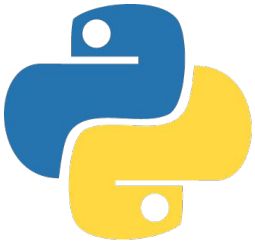
# Aplicaciones de escritorio y tkinter

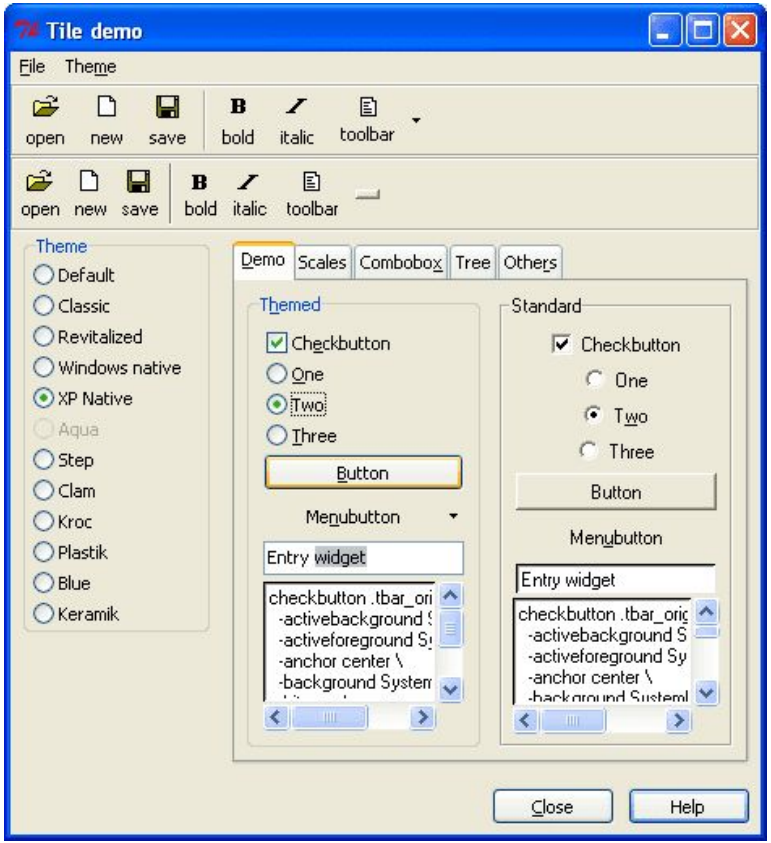
Python es siempre una buena elección para desarrollar aplicaciones de escritorio. Ofrece una gran cantidad de librerías para diseñar y desplegar interfaces de usuario rápidamente. Entre ellas se encuentran Tcl/Tk, GTK+, Qt y wxWidgets. Estas son librerías gráficas de código abierto y multiplataforma desarrolladas en C y C++, pero como Python permite escribir extensiones en dichos lenguajes y luego importarlas como módulos convencionales, utilizarlas es realmente muy fácil.

En este ocasión al igual que en el curso introductorio estaremos trabajando con Tcl/Tk (`tkinter`), pero con una profundidad aun mayor. Esta es una librería mucho más pequeña en comparación con otras, pero lo suficiente para tener una aplicación de escritorio funcional en muy pocos minutos y perfectamente preparada para aplicaciones comerciales. Además, accedemos a ella a través del módulo `tkinter`, módulo que viene en la librería estándar de Python, por lo que no requiere instalaciones adicionales.

Es importante aprender a desarrollar aplicaciones gráficas “manualmente” (esto es, escribiendo el código directamente).

Por otro lado, en el caso de librerías más grandes como GTK+ o Qt, hacer uso de dichas herramientas (como Glade o QtDesigner) requiere de un conocimiento previo de su funcionamiento para no terminar convirtiéndose en un impedimento para el desarrollo de la aplicación.






En la imagen observamos una aplicación escrita en Python usando Tcl/Tk. Destacamos distintos controles que incluye (botones, cajas de texto, etiquetas, menúes, etc.), a los cuales también se conoce como widgets. ¡Incluso la misma ventana es un *widget!*

El módulo `tkinter` está estructurado siguiendo el paradigma de orientación a objetos. Recordamos que el concepto central de este paradigma es el de clase, distinguimos por la convención de nombramiento que lleva CamelCase en lugar de lower_case. Por ejemplo:

```
a = Clase()
```

En la terminología de este paradigma, en el código anterior `Clase` es propiamente el nombre de una clase y `a` es una *instancia* de `Clase`. Puesto que cada *widget* de Tcl/Tk es una clase, tendrán nombres que sigan esa convención.

# ¡Sigamos trabajando!