این مار خوش خط و خال

قسمت پنجم

آشنایی و کار با زبان برنامه‌نویسی پایتون

» احمد شریف پور

بسیاری از کاربران معمول کامپیوتر، به‌ویژه تازه‌واردان دنیای لینوکس، به خط فرمان به دیده امری عجیب و ترسناک می‌نگرند که به کارگیری آن و تسلط بر آن تنها از عهده خبرگان و کهنه‌کاران دنیای کامپیوتر برمی‌آید. این کاربران و حتی بسیاری از استفاده‌کنندگان حرفه‌ای کامپیوتر ترجیح می‌دهند به جای تایپ دستورات و تعامل از طریق «متن»، دکمه‌ای را کلیک کرده، اسلایدری را جابه‌جا کرده یا پنجره‌ای را باز و بسته کنند. به عبارت ساده‌تر آن‌ها ترجیح می‌دهند که با یک رابط بصری سروکار داشته باشند تا یک رابط متنی. آن‌ها GUI را بیشتر می‌پسندند. به همین دلیل و بر اساس وعده‌ای که در شماره پیشین داده بودیم، در این شماره نحوه پیاده‌سازی GUI از طریق زبان پایتون را بررسی می‌کنیم.

کدام جعبه ابزار؟

برای ایجاد رابط‌های گرافیکی در پایتون، گزینه‌های متعددی وجود دارد که با توجه به سکوی مورد استفاده، محیط گرافیکی موردنظر و حتی کاربرد مورد نیاز می‌توانید از بین آن‌ها گزینه مناسب را انتخاب کنید. در ادامه چند نمونه از Toolkit یا جعبه ابزارهای مشهور ساخت رابط بصری بر مبنای پایتون را معرفی می‌کنیم.

• **TK Inter:** ساده‌ترین و در عین حال در دسترس‌ترین جعبه ابزار ساخت رابط‌های بصری است که به طور معمول به همراه غالب بسته‌های پایتون و به‌صورت پیش‌فرض نصب می‌شود. ویرایشگر IDLE به‌وسیله همین ابزار ساخته شده، اما سادگی و ظاهر نه چندان زیبای آن، به کاهش علاقه به استفاده از این جعبه ابزار منجر شده است.

• **PyGTK:** این جعبه ابزار در واقع wrapper یا پوسته‌ای است که به دور ابزار گرافیکی +GTK پیچیده می‌شود تا آن را برای پایتون قابل استفاده سازد. ابزار +GTK خود مادر تمام برنامه‌ها و پنجره‌های میزکار Gnome است و برنامه‌های ساخته شده با PyGTK هم دقیقاً ظاهری همانند برنامه‌های میزکار Gnome خواهد داشت. برای این جعبه ابزار، IDE خاصی وجود ندارد، اما طراحی ظاهر برنامه و فرم‌های آن بر مبنای این جعبه ابزار (بدون امکان ویرایش یا ایجاد کد) توسط برنامه‌هایی نظیر Glade و Gazpacho در میزکار Gnome امکان‌پذیر است.

• **PyQt:** این جعبه ابزار نیز wrapper یا پوسته‌ای است که به دور ابزار گرافیکی Qt پیچیده می‌شود تا آن را برای پایتون قابل استفاده سازد. ابزار Qt ابزاری است که کل میزکار KDE بر مبنای آن بنا شده است و برنامه‌های تولید شده با PyQT نیز دقیقاً ظاهری همانند برنامه‌های میزکار KDE خواهد داشت. برای کار با این جعبه ابزار می‌توانید از IDE قدرتمند eric در میزکار KDE استفاده کنید.

نام ToolKit	مزایا	معایب
TkInter	به همراه غالب نسخه‌های پایتون نصب می‌شود و توسعه رابط بصری با آن بسیار ساده است.	تعداد ابزارک‌ها یا ویجت‌های آن کمتر از سایر بسته‌ها است. رابط‌های حاصل از آن زیبایی چندانی ندارد.
PyGTK	براساس کتابخانه توابع GTK+ نوشته شده و برای توسعه در محیط‌هایی نظیر Gnome بسیار مناسب است. ابزارهای مناسب زیادی برای طراحی رابط بصری با استفاده از آن وجود دارد. ظاهر آن همانند تمام برنامه‌های GTK است.	علاوه بر پایتون، به نصب بسته PyGTK و ابزارهای مرتبط با آن نیاز دارد. ظاهر رابط‌های ساخته شده با آن، با محیط‌هایی نظیر KDE چندان همخوان نیست.
PyQt	براساس کتابخانه توابع Qt نوشته شده و برای توسعه در محیط KDE بسیار مناسب است. یک IDE تمام عیار با نام Eric برای آن موجود است.	علاوه بر پایتون، به نصب بسته PyQt و ابزارهای مرتبط با آن نیاز دارد. ظاهر رابط‌های ساخته شده با آن، با محیط‌هایی نظیر Gnome چندان همخوان نیست.
wxPython	براساس کتابخانه WxWidgets نوشته شده، کاملاً روی تمام پلتفرم‌های ویندوز و لینوکس پشتیبانی می‌شود. در تمام محیط‌ها ظاهر آن همانند نرم‌افزارهای بومی همان پلتفرم دیده خواهد شد. در لینوکس یک IDE مناسب به نام Boa Constructor دارد.	علاوه بر پایتون، به نصب بسته WxPython و ابزارهای مرتبط با آن نیاز دارد. پیاده‌سازی اجزای رابط کاربر بیش از سایر ابزارها به کدنویسی نیاز دارد.

جدول۱ مقایسه سریع ToolKitهای توسعه رابط بصری برای پایتون

• **wxPython:** این جعبه ابزار براساس wxWidget ساخته شده و یکی از قدرتمندترین جعبه ابزارهای طراحی رابط بصری است. مهم‌ترین مزیت آن مستقل بودن از پلتفرم و ظاهر بومی برنامه‌های تولید شده با آن است. IDE قدرتمند Boa Constructor یکی از بهترین گزینه‌ها برای توسعه رابط بصری کاربر با زبان پایتون است که در مطالب این قسمت و قسمت بعدی نیز از همین برنامه استفاده خواهد شد.

برای مقایسه سریع جعبه ابزارهای ساخت رابط گرافیکی می‌توانید به جدول ۱ مراجعه کنید.

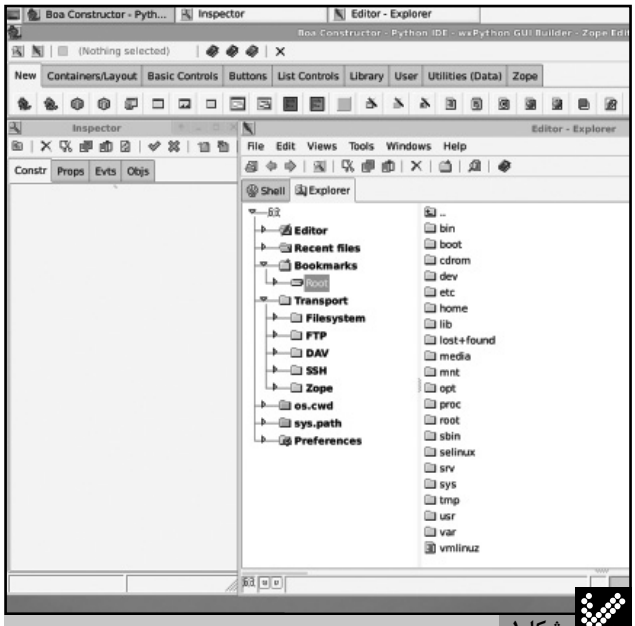
نصب ابزارها

همان‌طور که به احتمال متوجه شده‌اید، برای ادامه این مبحث به Boa Constructor و WxPython نیاز خواهیم داشت. برای نصب آن‌ها به Software Center اوبونتو رفته و از طریق بخش‌های Developer Tools و Graphic Interface Design نرم‌افزار Boa Constructor را نصب کنید. معمولاً انتخاب این بسته برای نصب، به نصب خود به خود WxPython نیز منجر خواهد شد. همچنین می‌توانید در خط فرمان دستورات زیر را تایپ کنید:

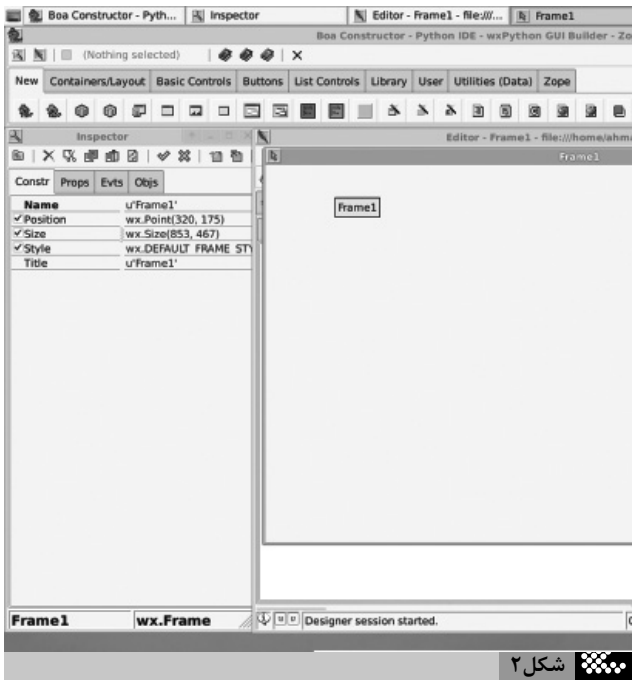
```
sudo apt-get install boa-constructor
sudo apt-get install python-wxversion
```

در هر دو حالت پس از وارد کردن گذرواژه سیستم، دانلود و نصب نرم‌افزارها آغاز خواهد شد. پس از اتمام کار نصب از طریق منوی Application، سپس گزینه Programming برنامه Boa Constructor را اجرا کنید. ظاهر برنامه پس از اجرا همانند شکل ۱ خواهد بود.

همان‌گونه که مشاهده می‌کنید این برنامه از سه پنجره یا به اصطلاح فریم تشکیل شده است. قاب یا فریم بالایی قاب ابزار یا Tool Frame نامیده می‌شود. قاب سمت چپ پایینی، قاب بازرسی یا Inspector Frame و قاب سمت راست پایین، قاب ویرایشگر یا Editor Frame است. قاب ابزار زبانه‌های متعددی مانند New، Container/Layout و... دارد. از این قاب می‌توانید برای ایجاد پروژه‌های جدید، افزودن قاب یا پنجره به پروژه‌های موجود یا افزودن کنترل‌های مختلف به قاب‌ها استفاده کنید. قاب بازرسی پس از افزودن قاب‌ها و کنترل‌های مختلف به پروژه بسیار مورد استفاده قرار خواهد گرفت. به‌صورت معمول این قاب ساختار درختی و روابط بین کنترل‌های مختلف را به ما نشان خواهد داد و در نهایت قاب ویرایشگر، امکان ویرایش کد، ذخیره پروژه و کارهایی از این قبیل را فراهم خواهد کرد.



شکل۱



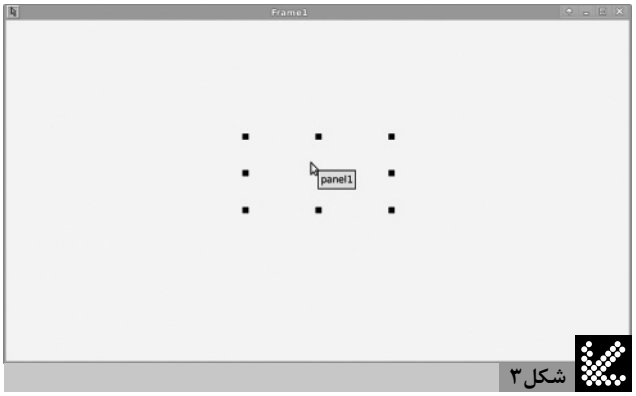
شکل۲

برای شـــروع کار ما ابتدا به سراغ قاب ابزارها خواهیم رفت. در زبانه New گزینه‌های مختلفی برای شـــروع کار وجود دارد اما در این بخش ما تنها از دکمه‌های پنجم و ششم، یعنی wx.App و wx.Frame استفاده خواهیم کرد. آیتم نخســـت، یعنی wx.App امکان ساخت یک پروژه کامل را فراهم می‌کند که از دو فایل Frame و Application تشـــکیل شده است و خود Boa این دو فایل را نیز به‌صورت خودکار تولید می‌کند. دکمه دوم یا wx.Frame امکان افـــزودن یک فرم به پروژه موجود را فراهم می‌کند. پر کاربردترین ابزارها در زبانه دوم یا Container/Layout ابزار wx.Panel (اولین آیتم از ســـمت چـــپ) و انواع مختلف کنترل کننده اندازه یا Sizer (دکمه‌های دوم تا ششـــم از سمت راســـت) هستند. در زبانه Basic Controls شما کنترل‌های متداول نظیر جعبه متن، جعبه چک، کنترل‌های لیست و... را مشاهده خواهید کرد و زبانه Utilities نیز برای دسترســـی به ابزارهای منو و تایمر مورد استفاده قرار خواهد گرفت.

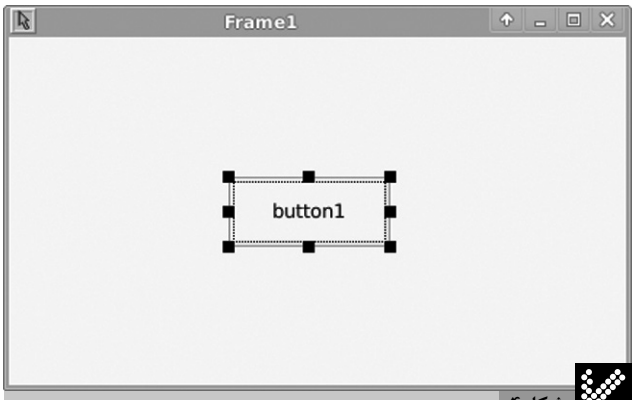
قبل از شـــروع کار با Boa لازم اســـت به دو نکته کوچک، اما مهم توجه کنید. نخســـت این‌کـــه برخی کنترل‌ها در پنجره طراحی یـــا Designer قابل جابه‌جایی نیستند. در صورت روبه‌رو شدن با چنین مشکلی، پس از انتخاب آن‌هـــا از ترکیب کلیدهای Ctrl و جهت‌نما برای حرکت دادن آن‌ها اســـتفاده کنیـــد. نکته دیگر این‌که هنگام قرار دادن یک کنترل از جنس panel روی یک فریم، پنل به ســـادگی دیده نخواهد شـــد. در این‌صورت به دنبال مربع‌های کوچک سیاه رنگی روی فریم بگردید یا از طریق قاب بازرسی و زبانه اشیا یا Objs این کنترل را انتخاب کنید.

شروع کار

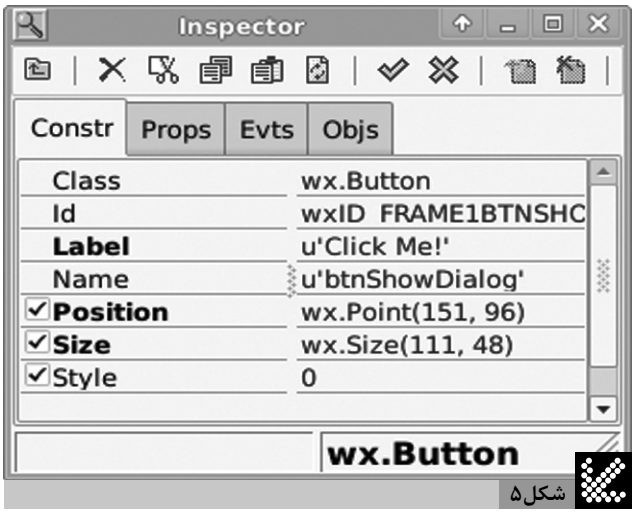
برای مرحله اول، در قاب ابزار و در زبانه New دکمه wx.App را کلیک کنید تا یک پروژه جدید ایجاد شـــود. در این حالت دو زبانه جدید یکی با نام *(App1)* دیگری با نام *(Frame1)* به قاب ویرایشگر شما اضافه خواهد شد. مناســـب‌ترین اقدام در این زمان ذخیره کردن پروژه ایجاد شده است. ایـــن کار را از زبانـــه *(Frame1)* آغاز می‌کنیم. در قاب ویرایشـــگر، دکمه Save را بزنید و این فایل را در یک پوشـــه دلخواه با نام Frame1.py ذخیره کنید. همان‌طور که مشاهده می‌کنید نام زبانه به Frame1 تغییر خواهد کرد. علامت‌های *()* به معنای ذخیره نشـــدن تغییرات زبانه جاری است. همین کار را برای زبانه *(App1)* نیز انجام دهید. در قاب ویرایشـــگر به غیر از دکمـــه Save چندیـــن دکمه مهم دیگر نیز وجـــود دارد که یکی از آن‌ها دکمه Run Application (به شـــکل یک مثلث زرد رنگ) برای اجرای برنامه است. در حال حاضر، حتی بدون یک خط کدنویســـی نیز برنامه شـــما قابل اجرا اســـت. با زدن همین دکمه شما می‌توانید نتیجه اجرای برنامه (که یک پنجره ساده و خالی است) را ببینید. در صورتی‌که شما در حال ویرایش یک فایل مربـــوط به یک قاب یا فریم باشـــید (همانند Frame 1) دکمه‌های دیگری نیز به نوار ابزارها افزوده می‌شـــود که مهم‌ترین آن‌ها دکمه Frame Designer اســـت (برای یافتن نام هر یک از دکمه‌ها، نشـــانگر ماوس را روی آن‌ها برده و کمی صبر کنید). فشـــردن این دکمه شـــما را به حالـــت طراحی می‌برد که در ایـــن وضعیـــت می‌توانید ظاهر فرم موردنظر را به دلخواه و با اســـتفاده از ابزارهـــای گرافیکـــی تغییر دهید. روی این دکمـــه کلیک کنید تا با ظاهری همانند شـــکل ۲ روبه‌رو شوید. فرم نمایش داده شده در واقع بستری است که می‌توانید کنترل‌های دلخواه خود را به آن اضافه کنید. نخســـتین آیتمی که باید به فرم اضافه کنید، یک کنترل از نوع پنل اســـت. یک رســـم نانوشته در توســـعه رابط‌های گرافیکی وجود دارد که شـــما را از قرار دادن مستقیم هر کنترلی به‌جز پنل روی یک فریم منع می‌کند. پس در قاب ابزارها به زبانه Container/Layout بروید. روی دکمه wx.Panel و سپس در محل دلخواهی از فریم کلیک کنید. اگر کار را به درســـتی انجام داده باشـــید، فریم ۱ باید به صورت شـــکل ۳ دارای تعدادی مربع ســـیاه رنگ کوچک باشـــد. همان‌طور



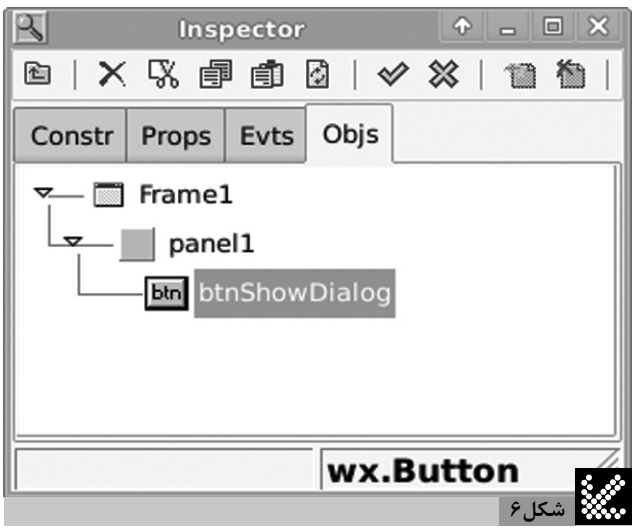
شکل ۳



شکل ۴



شکل ۵



شکل ۶

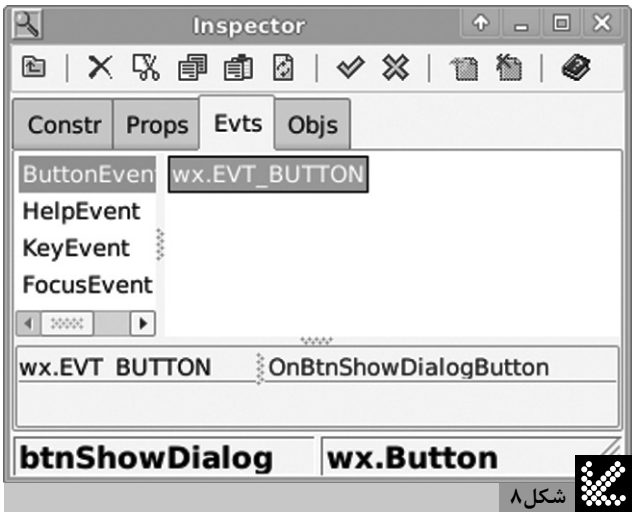
که پیش‌تر هم هشـــدار داده بودیم، خطوط بدنه پنل دیده نمی‌شـــود و برای ویرایش و جابه‌جایی آن باید به دنبال این ۸ مربع کوچک بگردید. برای تغییر اندازه پنل می‌توانید از این مربع‌ها استفاده کنید. در این تمرین ما می‌خواهیم پنل تمام ســـطح فریم را پوشـــش دهد. فریم را کمی کوچک‌تر و پنل را اندکی بزرگ‌تر کنید تا پنل تمام ســـطح فریم را پوشـــش دهد. حال پنلی در اختیار داریم که ســـایر کنترل‌ها روی آن قرار داده خواهند شـــد. اگر فریم را کمی جابـــه جا کنید، خواهید دید که دو دکمه جدید به نوار ابزار قاب ویرایشـــگر اضافه شـــده اســـت. یکی به شـــکل یک تیک و با نام Post و دیگری به شکل ضربدر با نام Cancel. کلیک دکمه Post باعث می‌شـــود تا از حالت Design خارج شـــوید و تغییرات اعمال شـــده در فریم به صورت کدهای پایتون به فایل آن منتقل شود. البته هنوز وظیفه ذخیره کردن فایل به عهده خود شما اســـت. دکمه Post دیگری هم روی قاب بازرســـی دیده می‌شود که بعدها به آن خواهیـــم پرداخـــت. دکمه Post را کلیک کنیدو پس از آن دوباره به حالت Design برگردید و در قاب ابزارها به زبانه دکمه‌ها یا Buttons مراجعه کنید. روی نخســـتین دکمه سمت راست (wx.Button) کلیک و آن را جایی تقریباً در مرکز فریم قرار دهید. در این حالت فریم شـــما باید شـــبیه شکل ۴ باشد. همانند پنل، ۸ مربع کوچک برای تغییر اندازه دکمه در اطراف آن وجود دارد. اگر برای جابه‌جا کردن دکمه با مشکل مواجه شدید، براساس توضیح داده شده، از ترکیب کلیدهای Ctrl و جهت‌نما استفاده کنید.

اکنون زمان آن رســـیده که به ســـراغ قاب بازرسی برویم. در حالی‌که کنتـــرل دکمه در حالت انتخاب اســـت، به ســـراغ این قـــاب و زبانه Constr می‌رویم. در این قسمت می‌توانید مشخصات و خواص کنترل انتخاب شده را ویرایـــش کنید. فعلاً نام کنترل دکمه را به btnShowDialog و برچســـب یـــا Label آن را بـــه !Click Me تغییر دهید. ظاهـــر زبانه Constr باید همانند شـــکل ۵ باشـــد. ما به بقیه تنظیمات این زبانه کاری نداریم و به طور مستقیم به ســـراغ زبانه Objs می‌رویم. در این زبانه شـــما تمام کنترل‌های استفاده شـــده و روابط والد و فرزندی آن‌ها را مشـــاهده خواهید کرد. همان‌گونه که در شـــکل ۶ مشاهده می‌کنید، کنترل دکمه فرزند پنل و پنل خود فرزند فریم اســـت. با زدن دکمه Post در قاب بازرسی تغییرات را به فایل اعمال کرده و آن را ذخیره کنید. دوباره به حالت Design بازگشته و به زبانه Objs از قاب بازرسی مراجعه کنید. اگر فریم انتخاب نشده است آن را انتخاب کرده و به زبانـــه Constr بروید. در آنجا عنـــوان فریم را به My First GUI تغییر دهید. تغییرات را Post کرده، سپس فایل را ذخیره کنید.

در ادامـــه با اســـتفاده از مثلث زرد رنـــگ Run Application برنامه را اجرا کنید. فرمی همانند شـــکل ۷ ظاهر خواهد شـــد، اما کلیک دکمه وســـط فـــرم، هیچ عکس‌العملی ایجاد نخواهد کـــرد، زیرا ما هنوز هیچ عملکردی را بـــه آن دکمه نســـبت نداده‌ایم. برای این کار ما باید یـــک رویداد یا Event را تعریف کرده و به دکمه نسبت بدهیم. پنجره فعلی را بسته و دوباره به حالت Design بروید. پس از کلیک و انتخاب دکمه، به زبانه Evts در قاب بازرسی مراجعه کنید. از لیست سمت چپ ButtonEvent را انتخاب کرده و از لیست ســـمت راســـت wx.EVT_BUTTON را دوبار کلیک کنیـــد. همان‌گونه که در شکل ۸ مشاهده می‌کنید، در زیر پنجره رویدادی با نام OnBtnShowDialog ساخته شده است. تغییرات را Post و ذخیره کنید. در پایین‌ترین قسمت قاب ویرایشـــگر باید تابعی با نام OnBtnShowDialog ساخته شده باشد. کاری که اکنون می‌خواهیم انجام دهیم، فراخوانی یک جعبه متن یا MessageBox اســـت کـــه متنی را برای ما بـــه نمایش درآورد. در اینجا مـــا از تابع داخلی wx.MessageBox اســـتفاده می‌کنیم. در قاب ویرایشـــگر کدهای زیر را در یک خط جایگزین ()event.Skip کنید:

```
wx.MessageBox("You clicked the button !" ,
"Info",wx.ICON_QUESTION)
```

شکل ۷



شکل ۸

شکل ۹

تغییـــرات را ذخیـــره کـــرده و با مثلـــث زردرنگ برنامـــه را اجرا کنید. ایـــن بار کلیک دکمه وســـط فرم بایـــد پیغامی همانند شـــکل ۹ را به نمایش درآورد. توجه کنید که آیکون علامت سؤال ممکن است با توجه به تنظیمات ظاهر سیســـتم شـــما شـــکل متفاوتی داشته باشـــد. در تابع MessageBox ســـه آرگومان وارد شـــده به ترتیـــب عبارتند از، متن پیغـــام، عنوان پنجره پیغـــام و نـــوع آیکـــون مـــورد اســـتفاده. برای تنظیـــم آیکـــون گزینه‌های دیگری نظیـــر wx.ICON_EXCLAMATION ،wx.ICON_INFORMATION و wx.ICON_ERROR را نیـــز در اختیـــار دارید که می‌توانیـــد آن‌ها را نیز به سادگی امتحان کنید.

ایـــن مثال در عین ســـادگی، نخســـتین برنامه کامل شـــما بـــر مبنای یـــک رابـــط بصری بـــود. در قســـمت بعـــدی بـــا امکانـــات wxPython و Boa Constructor بیشـــتر آشـــنا خواهیم شد و نمونه‌های پیچیده‌تری را پیاده خواهیم کرد.