طراحی رابط کاربری با PyQt در پایتون

تألیف:

دکتر رمضان عباس نژادورزی
مهندس نوراله کریم‌تبار احمدچالی

فن‌آوری نوین

سرشناسه	:	عباس‌نژاد ورزی، رمضان، ۱۳۴۸ -
عنوان و نام پدیدآور	:	طراحی رابط کاربری با PyQt در پایتون/ تالیف رمضان عباس‌نژادورزی، نوراله کریم‌تباراحمدچالی.
مشخصات نشر	:	بابل: فناوری نوین، ۱۳۹۸.
مشخصات ظاهری	:	۱۴۸ ص.: مصور ،جدول.
شابک	:	۳۵۰۰۰۰ ریال: 978-600-7272-31-2
وضعیت فهرست نویسی	:	فیپا
موضوع	:	کیوت (منبع الکترونیکی)
موضوع	:	Qt (Electronic resource)
موضوع	:	پیتون (زبان برنامه‌نویسی کامپیوتر)
موضوع	:	Python (Computer program language)
موضوع	:	رابط نرم‌افزاری گرافیکی
موضوع	:	Graphical user interfaces (Computer systems)
موضوع	:	نرم‌افزار کاربردی -- طراحی و توسعه
موضوع	:	Application software – Development

w

ww.fanavarienovin.net

فـــن‌آوری نـــوین بابل، کد پستی ۴۷۱۶۷-۷۳۴۴۸ تلفن: ۳۲۲۵۶۶۸۷-۰۱۱

طراحی رابط کاربری با PyQt در پایتون

تألیف: رمضان عباس نژادورزی – نوراله کریم‌تبار احمدچالی

نوبت چاپ: چاپ اول

سال چاپ: تابستان ۱۳۹۸

شمارگان: ۱۰۰

قیمت: ۳۵۰۰۰ تومان

نام چاپخانه و صحافی: دفتر فنی سورنا

شابک: ۹۷۸-۶۰۰-۷۲۷۲-۳۱-۲

نشانی ناشر: بابل، چهارراه نواب، کاظم بیگی، جنب مسجد منصور کاظم بیگی، طبقه اول

طراح جلد: کانون آگهی و تبلیغات آبان (احمد فرجی)

تهران، خ اردیبهشت، نبش وحید نظری، پلاک ۱۴۲ تلفکس: ۶۶۴۰۰۲۲۰-۶۶۴۰۰۱۴۴

فهرست مطالب

فصل اول: آشنایی با PyQt

۷ .....

۱-۱. رابط کاربری..... ۷

۱-۲. مروری بر API PyQt..... ۷

۱-۳. مفاهیم اولیه در PyQt..... ۷

۱-۳-۱. پنجره ۷

۱-۳-۲. ویجت‌ها..... ۷

۱-۳-۳. مفهوم سیگنال و اسلات..... ۸

۱-۴. اضافه کردن ماژول PyQt به برنامه ۸

۱-۵. اضافه کردن پنجره..... ۱۹

۱-۶. ویجت QLabel..... ۱۹

۱-۷. ویجت QLineEdit ۲۶

۱-۸. مدیریت طرح‌بندی..... ۳۰

۱-۹. کنترل QPushButton..... ۴۱

فصل دوم: برخی از ویجت‌های مهم PyQt

۵۰ .....

۲-۱. ویجت QCheckBox..... ۵۰

۲-۲. ویجت QListWidget..... ۵۶

۲-۳. ویجت QComboBox..... ۵۹

۲-۴. ویجت QProgressBar..... ۶۳

۲-۵. ویجت QSpinBox..... ۶۷

۲-۶. ویجت QSlider..... ۷۰

۲-۷. ویجت QMenu..... ۷۰

۲-۸. ویجت QToolBar ۷۰

۲-۹. ویجت QStatusBar..... ۷۰

۲-۱۰. ویجت QTabWidget..... ۷۰

۲-۱۱. ویجت QStackedWidget..... ۷۰

۲-۱۲. ویجت QDockWidget..... ۷۰

فصل سوم: مباحث پیشرفته در PyQt ۷۲

۳-۱. برنامه‌های چند پنجره‌ای..... ۷۲

۳-۲. کشیدن و رها کردن (Drag and Drop)... ۷۳

۳-۳. گرافیک در PyQt

..... ۸۱

۳-۴. کلاس QClipboard..... ۸۶

فصل چهارم: کادرهای محاوره در PyQt

۹۰ .....

۴-۱. کلاس QMessageBox..... ۹۰

۴-۲. کلاس QinputDialog..... ۹۱

۴-۳. کلاس QFontDialog..... ۹۹

۴-۴. کلاس QColorDialog..... ۱۰۳

۴-۵. کلاس QFileDialog..... ۱۰۴

فصل پنجم: طراح Qt ۱۱۴

۵-۱. ابزار Qt Designer..... ۱۱۴

۵-۲. اضافه کردن ویجت..... ۱۱۹

۵-۳. اضافه کردن ویجت..... ۱۲۰

۵-4. حذف ویجت‌ها از پنجره..... ۱۲۰

۵-4. تغییر خواص ویجت‌ها..... ۱۲۰

۵-5. افزودن Signals & Slots..... 133

۵-6. روند ساخت یک برنامه با PyQt ۱۲۰

منابع: ۱۴۸

مقدمه

زبان برنامه‌نویسی پایتون کاربردهای متعددی دارد. از طریق این زبان می‌توانید برنامه‌های تحت کنسول، برنامه‌های تحت ویندوز(دسک‌تاپی)، برنامه‌های تحت وب، برنامه‌های تحت شبکه و غیره را پیاده‌سازی نمایید. پایتون برای پیاده‌سازی این برنامه‌ها از کتابخانه‌های مختلفی بهره می‌برد. یکی از کتابخانه‌های معروف پایتون که برای **طراحی و پیاده‌سازی رابط گرافیکی کاربر** به کار می‌رود، **کتابخانه PyQt** است. قبل از مطالعه این کتاب باید مقدمات برنامه‌نویسی پایتون یاد بگیرید. برای این منظور می‌توانید دو کتاب به نام‌های **آموزش گام به گام برنامه‌نویسی پایتون** و **حل مسائل پایتون** از همین انتشارات را مطالعه کنید.

کتابخانه PyQt در اصل از QT گرفته شده است. این کتابخانه امکانی را فراهم می‌کند تا برنامه-نویس بدون نیاز به کتاب‌خانه‌ها و ابزارهای جانبی دیگر **به تولید رابط‌های گرافیکی (بصری) کاربر** بپردازد. این کتاب کتابخانه PyQt را به طور کامل آموزش می‌دهد. کتاب حاضر شامل 5 فصل و یک پیوست در کتاب الکترونیکی است. **فصل اول**، مقدمات PyQt، ویجت‌های QLabel، QLineEdit، QPushButton را با مثال‌های متعدد آموزش می‌دهد. **فصل دوم**، ویجت‌های دیگر مانند QCheckBox، QListBox، ComboBoxQ، QMenu و غیره را مورد بررسی قرار می‌دهد. **فصل سوم**، مباحث پیشرفته PyQt را آموزش می‌دهد. **فصل چهارم**، کادرهای محاوره را به طور کامل بررسی می‌کند. **فصل پنجم**، طراح Qt را بحث می‌نماید. پیوست کتاب که در کتاب الکترونیکی آمده است شامل خلاصه‌ای از آموزش مقدمات پایتون در پایتون است.

امیدواریم این کتاب نیز مورد استقبال اساتید و دانشجویان رشته‌های مختلف که زبان پایتون را مطالعه می‌کنند، قرار گیرد.

مولفین
fanavarienovin@gmail.com

فصل ۱ آشنایی با PyQt

۱–۱. رابط کاربری

یک رابط کاربری می‌تواند مجموعه‌ای از دستورات یا تصاویر باشد. درصورتی‌که ارتباط کاربری به‌صورت دستور باشد، این ارتباط با درج دستورات توسط کاربر صورت می‌پذیرد و درصورتی‌که رابط به‌صورت تصویری باشد کاربر با انتخاب از میان چندین گزینه با برنامه ارتباط برقرار می‌کند.

رابط کاربری یکی از مهم‌ترین بخش‌های یک برنامه می‌باشد، چراکه مشخص‌کننده‌ی سهولت یا دشواری یک برنامه جهت انجام فرآیند موردنیاز کاربر است. نوع بصری رابط کاربری که با عنوان GUIs شناخته می‌شود و مخفف عبارت Graphical User Interface است، متداول‌ترین نوع رابط کاربری می‌باشد که تعامل با برنامه را بسیار آسان و جذاب می‌سازد.

تمرکز طراحان رابط کاربری بیش‌تر بر پیش‌بینی این موارد است که کاربر چه نیازهایی ممکن است داشته باشد و تلاش می‌کنند تا اطمینان حاصل کنند که رابط کاربری دارای عناصری است که دسترسی و فهمیدن آن‌ها برای کاربر آسان است.

به‌طورکلی UI شامل مفاهیمی چون طراحی تعاملی، طراحی بصری و معماری اطلاعات می‌باشد.

انتخاب عناصر رابط کاربری

طراحی رابط کاربری مناسب علاوه بر سرعت بخشیدن به فرآیند انجام کار و جلب رضایت مشتری باعث صرفه‌جویی در هزینه‌های کلی نیز خواهد شد.

به‌طورکلی عناصر رابط کاربری به‌صورت زیر دسته‌بندی می‌شوند:

- **کنترل‌های ورودی:** شامل دکمه‌ها، فیلدهای متنی، چک باکس‌ها، دکمه‌های رادیویی، فیلدهای داده و غیره.
- **بخش‌های ناوبری:** شامل اسلایدها، فیلدهای جست‌وجو، تگ‌ها، آیکون‌ها و غیره.
- **بخش‌های اطلاعاتی:** شامل جعبه‌های پیام، نوار پیشرفت (Progress bar)، آگاه‌سازی‌ها و غیره.

رابط کاربری نهایی ترکیبی از عناصر فوق می‌باشد. انتخاب نوع عناصر و چگونگی به‌کارگیری آن‌ها بسته به نوع رابط کاربری متفاوت خواهد بود و چیدمان مناسب عناصر بسیار حائز اهمیت می‌باشد.

نکاتی که می‌بایست در طراحی رابط کاربری مدنظر قرار گیرند

انتخاب نوع و روند طراحی رابط کاربری به شناخت طراح از کاربر شامل اهداف کاربران، مهارت‌های آن‌ها، ترجیحات و تمایلات آن‌ها بستگی دارد. با شناخت کافی از کاربران و در نظر گرفتن موارد زیر می‌توان رابط کاربری مناسبی طراحی کرد:

- **ساده نگه‌داشتن رابط:** بهترین رابط کاربری، رابطی است که علاوه بر جامعیت تا حد ممکن ساده باشد. حتی‌المقدور باید استفاده از عناصر مختلف را کاهش داد.
- **طراحی هدفمند صفحات:** باید در نظر داشت که رابطه بین آیتم‌های یک صفحه و ساختار محتوایی آن صفحه بسیار حائز اهمیت می‌باشد. جایگذاری صحیح آیتم‌ها می‌تواند توجه کاربر را به مهم‌ترین نقاط صفحه جلب کرده و از هدر رفت وقت جلوگیری کند.
- **استفاده استراتژیک از رنگ و بافت:** با استفاده از رنگ و نور و تضاد میان آن‌ها علاوه بر این که می‌توان توجه کاربران را به بخشی جلب کرد می‌توان توجه آن‌ها نسبت به بخش‌های غیرضروری را کم رنگ کرد.
- **استفاده از تایپوگرافی مناسب:** استفاده از فونت مناسب تأثیری زیادی در خوانایی محتوا دارد. بولد کردن متون در مواقع لزوم، استفاده از سایزهای مختلف فونت و غیره می‌تواند تمرکز کاربر هنگام مطالعه متون را افزایش دهد.

۲-۱. مروری بر API PyQt

PyQt یکی از محبوب‌ترین پیوندهای پایتون است. PyQt توسط Riverbank Computing Limited توسعه‌یافته است. Qt خود به‌عنوان بخشی از پروژه Qt توسعه داده‌شده است. PyQt۵ در ویندوز، لینوکس، مک OS و سیستم‌عامل‌های مختلف یونیکس اجرا می‌شود.

PyQt یک ابزارک ویجت Gui است که یک رابط کاربری پایتون برای qt / یکی از قدرتمندترین و محبوب‌ترین رابط‌های گرافیکی است.

آخرین نسخه آن را می‌توانید از سایت riverbankcomputing.com دانلود کنید.

این ماژول یک بخش قابل‌نصب و قابل‌استفاده‌ی مجدد است که برای نمایش محتوا یا انجام عملیات خاص استفاده می‌شود.

API PyQt، مجموعه بزرگی از کلاس‌ها و متدها است. این کلاس‌ها در بیش از ۲۰ ماژول تعریف‌شده‌اند. برخی از ماژول‌های اصلی PyQt که اغلب استفاده می‌شوند، عبارت‌اند از:

- **ماژول QtCore**، کلاس‌های غیر GUI هسته‌ای است که توسط ماژول‌های دیگر استفاده می‌شود. به‌عبارت‌دیگر، این ماژول شامل کلاس‌های غیر رابط گرافیکی کاربر، ازجمله حلقه رویداد و سیگنال‌های کیوت و مکانیزم حافظه است. همچنین شامل پلت فرم انتزاعی مستقل برای یونیکد، فایل‌های نقشه‌برداری، حافظه مشترک، عبارت منظم، و تنظیمات نرم‌افزار و کاربر می‌باشد.
- **ماژول QtGui**، اجزای رابط گرافیکی کاربر را تشکیل می‌دهد. این سرویس‌ها عبارت‌اند از: تعداد جدول، درخت و کلاس فهرست بر اساس الگوی طراحی کنترل- نمایش- ماژول است. همچنین ویجت canvas ۲D که قادر به ذخیره‌سازی هزاران اقلام ازجمله ویجت عادی ارائه شده‌است.
- **ماژول QtMultimedia**، شامل کلاس‌هایی برای برنامه‌نویسی چندرسانه‌ای سطح – پایین است.
- **ماژول QtNetwork**، شامل کلاس‌هایی برای برنامه‌نویسی شبکه است. این ماژول شامل کلاس‌هایی برای ساخت برنامه‌های سمت سرویس‌گیرنده یا سرویس‌دهنده بر پایه پروتکل‌های TCP یا UDP

می‌باشد. همچنین، این بخش شامل کلاس‌های سرویس‌گیرنده FTP, HTTP و همچنین جست‌وجوی DNS می‌باشد.

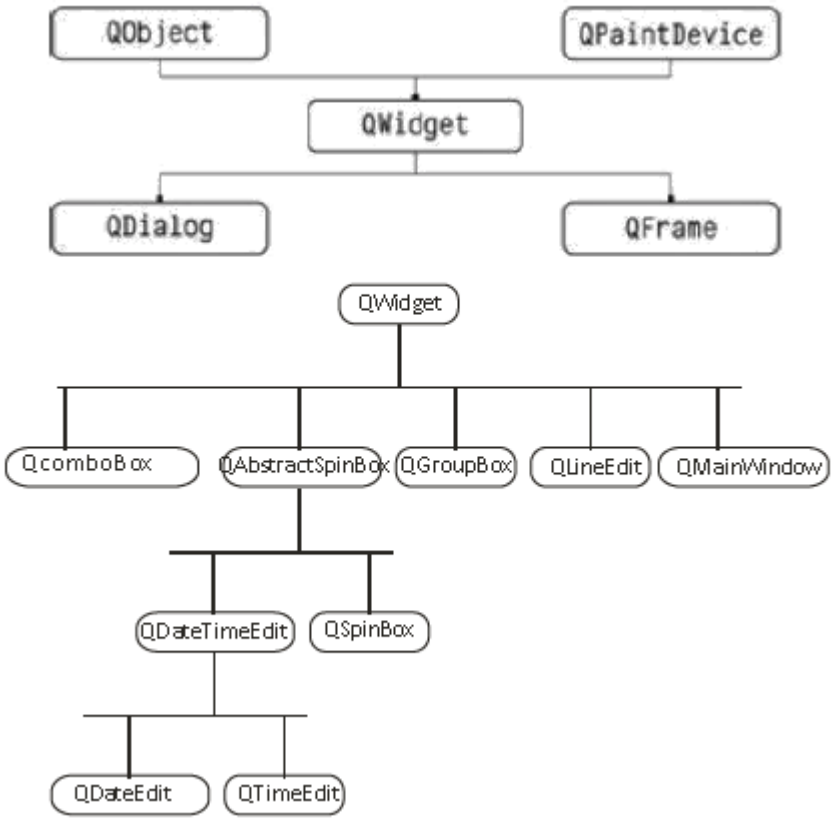
- **ماژول QtOpenGL**، کلاس‌های پشتیبانی OpenGL را برای پردازش تصویر تشکیل می‌دهد (این ماژول امکان ایجاد و کار بر روی تصاویر سه‌بعدی با استفاده از OpenGL را فراهم می‌کند.).
- **ماژول QtScript**، شامل کلاس‌هایی برای ارزیابی اسکریپت‌های Qt است. با استفاده از این ماژول برنامه‌های نوشته‌شده قادر به اجرای اسکریپت‌های جاوا خواهند بود. در حقیقت، با استفاده از این روش امکان توسعه و اسکریپت نویسی برای کاربران فراهم می‌شود.
- **ماژول QtSql**، کلاس‌هایی را شامل می‌شود که برای پرس‌وجو از پایگاه داده با استفاده از SQL به کار می‌رود. وظیفه این ماژول ایجاد ارتباط با پایگاه داده‌های مبتنی بر SQL می‌باشد. همچنین، این ماژول شامل مدیر پایگاه داده SQLite می‌باشد.
- **ماژول QtSvg**، شامل کلاس‌هایی برای نمایش محتویات فایل‌های SVG(Scalable Vector Graphics) می‌باشد.
- **ماژول QtWebKit**، از کلاس‌هایی برای رندر کردن و ویرایش HTML تشکیل‌شده است.
- **ماژول QtXml**، شامل کلاس‌هایی است که برای مدیریت XML به کار می‌روند.
- **ماژول QtAssistant**، شامل کلاس‌هایی است که برای پشتیبانی کمکی آنلاین استفاده می‌شوند. با استفاده از این ماژول می‌توان نرم‌افزار Qt Assistant را در برنامه نوشته‌شده با پایتون و Qt درونی سازی کرده و درنتیجه، مستندات برنامه را به همراه آن و بدون نیاز به ایجاد بخشی جداگانه در خود برنامه درونی سازی نمود.
- **ماژول QtDesigner**، کلاس‌های را شامل می‌شود که برای توسعه Qt Designer مورداستفاده قرار می‌گیرند. توسط این ماژول امکان توسعه طراح کیوتی توسط PyQt فراهم خواهد شد. برای مثال، می‌توان افزونه یا یک عنصر جدید در PyQt ایجاد و از آن همانند سایر اشیا گرافیکی (Widget) در محیط طراحی استفاده کرد.
- **ماژول AXContainerQ**، با استفاده از این ماژول برنامه نوشته‌شده قادر خواهد بود به اشیا COM و اکتیوایکس (ActiveX) ارتباط برقرار کند.
- **ماژول Qt**، شامل تمامی موارد ذکرشده در بالا می‌باشد. با اضافه کردن این ماژول دیگر نیازی نیست برنامه‌نویس ماژولی که تابع موردنیازش در آن قرار دارد را بداند. از معایب این روش لود شدن همه فریم ورک کیوتی می‌باشد که باعث مصرف فضای زیاد حافظه می‌گردد.
- **ماژول uic**، شامل کلاس‌هایی است که برای کار با فایل‌های ui می‌باشد و توسط طراح (Designer) کیوتی ایجاد می‌گردد.

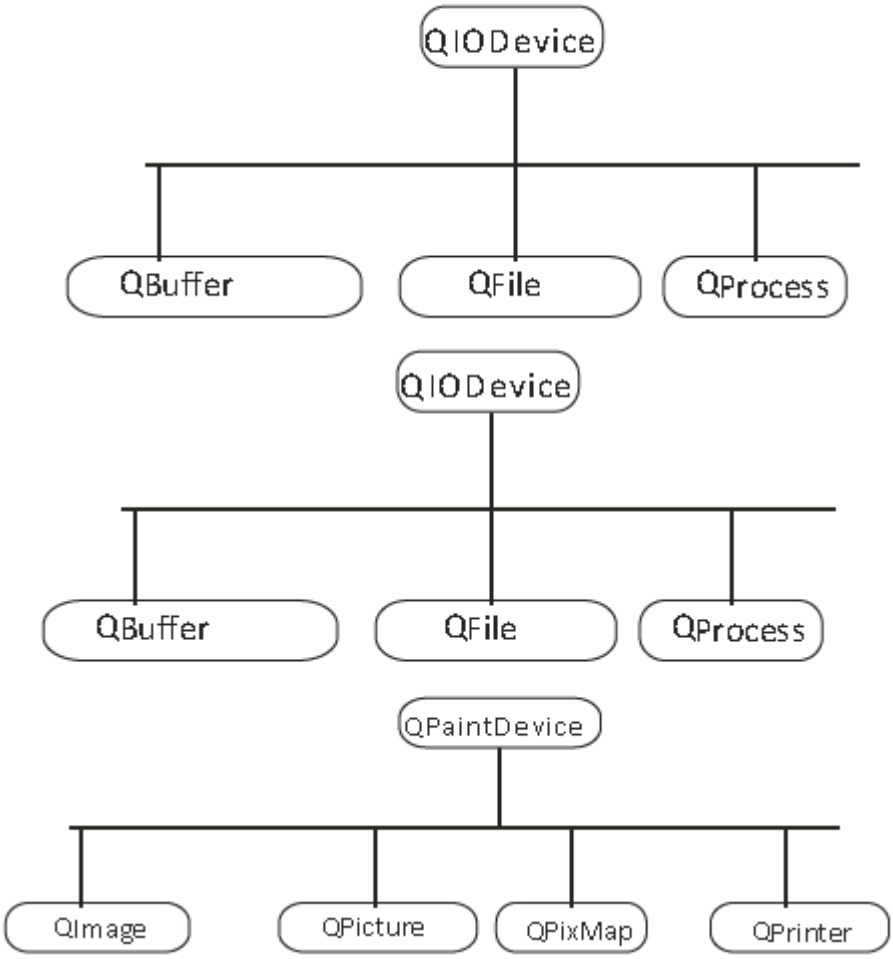
API PyQ، شامل بیش از ۴۰۰ کلاس است. کلاس QObject در بالای سلسله‌مراتب کلاس‌ها قرار دارد. این کلاس، پایه تمام اشیا Qt است. علاوه بر این، کلاس QPaintDevice، کلاس پایه‌ای برای تمام اشیایی است که می‌توان آن‌ها را نقاشی کرد.

کلاس QApplication، تنظیمات اصلی و جریان کنترل یک برنامه GUI را مدیریت می‌کند. این کلاس شامل حلقه رویداد اصلی است که رویدادهای ایجادشده توسط عناصر پنجره و سایر منابع پردازش و ارسال می‌شود. کلاس QApplication، همچنین تنظیمات گسترده سیستم و نرم‌افزار را کنترل می‌کند.

کلاس QWidget، از کلاس QObject و QPaintDevice مشتق شده است که کلاس پایه برای تمام اشیای رابط کاربر می‌باشد. کلاس QDialog و QFrame نیز از کلاس QWidget مشتق می‌شوند. این کلاس‌ها دارای سیستم کلاس فرعی خودشان هستند.

در شکل‌های زیر برخی از کلاس‌های مهم به‌صورت سلسله‌مراتبی نشان داده‌شده‌اند.





۳–۱. مفاهیم اولیه در PyQt

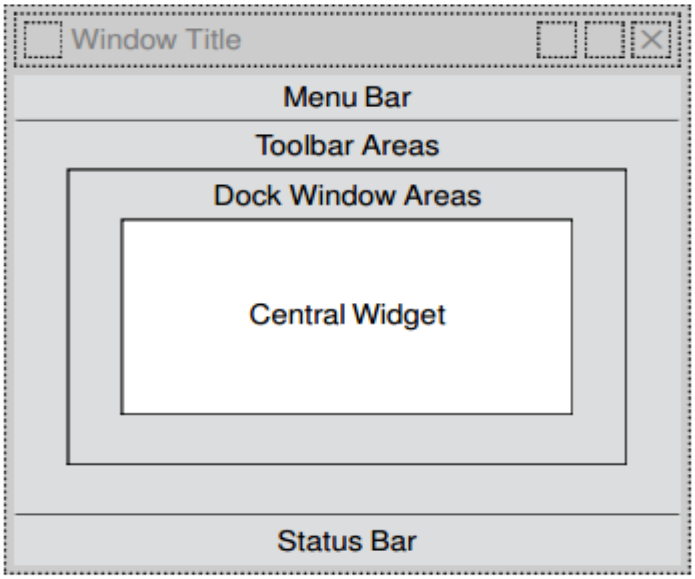
قبل شروع برنامه‌نویسی با PyQt با چند مفهوم رایج آشنا خواهیم شد. این مفاهیم عبارت‌اند از:

۱–۳–۱. پنجره

در طول دوره‌های مختلف تحصیلی، ممکن است با دوستان خود یک روزنامه دیواری درست کرده باشید. برای ساخت روزنامه دیواری ابتدا به یک مقوا نیاز دارید تا مطالب خود را بر روی آن بنویسید و یا عکس‌ها و بریده‌های مجلات را بر روی آن بچسبانید. در ساخت یک برنامه با واسط گرافیکی کاربر نیز باید صفحه یا پنجره‌ای(Window) در اختیار داشته باشید تا انواع دکمه‌ها، منوها، تصاویر و نوشته‌ها را روی آن قرار دهید. این پنجره، **واسط گرافیکی کاربر** (GUI) نامیده می‌شود. پنجره، صفحه‌ای است که اجزای گرافیکی گوناگونی بر روی آن قرار می‌گیرند. اندازه یک پنجره متناسب با تعداد و اندازه اشیای گرافیکی است که قرار است روی آن قرار گیرند. اصطلاح پنجره،

معانی متفاوتی در متون مختلف دارد، ولی در کل به یک محیط مستطیلی شکل اطلاق می‌گردد که بر روی صفحه‌نمایش قرار دارد.

یک پنجره GUI توسط شیء QMainWindowWidget ایجاد می‌شود. بعضی از ویجت‌ها در مکان خودشان در این پنجره اصلی قرار می‌گیرند، درحالی‌که ویجت‌های دیگر در منطقه مرکزی اضافه خواهند شد. شکل زیر فریم‌ورک QMainWindow را نشان می‌دهد:



۲-۳-۱. ویجت‌ها

PyQt ویجت‌های (**Widgets**) مختلفی مانند QLabel، QPushButton، QLineEdit، QCheckBox، QMenu و غیره را برای ایجاد برنامه‌های گرافیکی و رابط گرافیکی کاربر آماده کرده است. معمولاً ویجت، عنصر یا کنترل نام دارد. برخی از ویجت‌های پرکاربرد عبارت‌اند از:

- **ویجت QLabel**، به‌عنوان نگه‌دارنده عمل می‌کند تا متن یا تصویر غیرقابل‌ویرایش و یا یک فیلم **GIF** متحرک را نمایش دهد. همچنین، می‌تواند به‌عنوان یک کلید mnemonic برای ویجت‌های دیگر استفاده شود.
- **ویجت QLineEdit**، جعبه‌ای است که در آن یک خط از متن را می‌توان وارد کرد. برای واردکردن متن چند سطری، شیء **QTextEdit** موردنیاز است.
- **ویجت QPushButton**، یک دکمه ارائه می‌دهد تا وقتی کاربر آن را کلیک کند، یک تابع خاص را فراخوانی کند.
- **ویجت QRadioButton**، یک دکمه قابل انتخاب با یک برچسب متن را فراهم می‌کند. کاربر می‌تواند یکی از گزینه‌های ارائه‌شده در فرم را انتخاب کند. این کلاس از کلاس **QAbstractButton** مشتق شده است.

- **ویجت QCheckBox**، جعبه مستطیلی قابل انتخاب قبل از متن شیء **QCheckBox** قرار می‌گیرد تا کاربر بتواند گزینه‌های دو حالته را انتخاب نماید یا از حالت انتخاب خارج کند.
- **ویجت QComboBox**، لیستی از آیتم‌ها را دارد که فقط یک گزینه آن قابل انتخاب است.
- **ویجت QSpinBox**، یک جعبه متن به همراه یک عدد صحیح را نشان می‌دهد که کاربر می‌تواند با دکمه‌های بالا / پایین عدد موجود در کنترل را اضافه یا کم کند.
- **ویجت QSlider**، یک شیار را به همراه یک دسته نشان می‌دهد که کاربر می‌تواند از طریق دسته بر روی شیار حرکت کرده و مقدار آن را تعیین کند. این یک ویجت کلاسی برای کنترل مقادیر یک محدود به کار می‌رود.
- **ویجت QMenuBar**، **QMenu** و **QAction**، یک ویجت **QMenuBar** افقی در زیر نوار عنوان یک شیء **QMainWindow** قرار می‌گیرد تا اشیا **QMenu** را نمایش دهد.
- **ویجت QToolBar**، یک پانل متحرک ایجاد می‌کند که شامل دکمه‌های متنی، دکمه‌هایی با آیکون و یا ویجت‌های دیگر است.
- **ویجت QInputDialog**، شامل یک کادر محاوره پیش‌ساخته با یک فیلد متنی و دو دکمه **OK** و **Cancel** است.
- **ویجت QFontDialog**، کادر محاوره انتخاب فونت را نمایش می‌دهد تا کاربر بتواند فونت موردنظرش را انتخاب نماید.
- **ویجت QFileDialog**، یک کادر محاوره **انتخاب** فایل را نشان می‌دهد. این کادر محاوره کاربر را قادر می‌سازد تا بین فایل‌ها حرکت کند و یک فایل را برای باز کردن یا ذخیره انتخاب کند.
- **ویجت QTab**، اگر یک فرم دارای تعداد زیادی فیلد باشد که باید به‌طور هم‌زمان نمایش داده شوند، می‌توان آن‌ها را در صفحات مختلف قرار داده و هر صفحه را در یک ویجت **QTab** داد. **QTab**، نوار تب و یک صفحه را فراهم می‌کند.
- **ویجت QStacked**، عملکرد **QStackedWidget** شبیه به **QTabWidget** است. همچنین، در استفاده کارآمد از منطقه سرویس‌گیرنده پنجره کمک می‌کند.
- **ویجت QDock**، یک پنجره **dockable** یک پنجره فرعی است که می‌تواند در حالت شناور باقی بماند یا می‌تواند به پنجره اصلی در یک موقعیت مشخص متصل شود. شیء پنجره اصلی کلاس **QMainWindow** دارای یک منطقه برای پنجره‌های **dockable** است.
- **ویجت QStatusBar**، شیء **QMainWindow** یک نوار افقی به نام نوار وضعیت در پایین خودش دارد. این ویجت برای نمایش نوار وضعیت استفاده می‌شود.

- **ویجت QListWidget**، یک رابط مبتنی بر آیتم، برای اضافه کردن یا حذف گزینه‌ها از یک لیست است. هر آیتم در لیست یک شی ء **QListWidgetItem** است. **ListWidget** می‌تواند مجموعه‌ای از گزینه‌هایی باشد که چند گزینه آن قابل انتخاب است.
- **ویجت QScrollBar**،کنترل اسکرول یا نوار جابه‌جایی را اضافه می‌کند. اسکرول اجازه می‌دهد تا کاربر بتواند به بخش‌های از سند که قابل‌رؤیت نیستند، دسترسی یابد.
- **ویجت QCalendar**، یک **کنترل** تقویم را به پنجره اضافه می‌کند. کاربر می‌تواند تاریخ را با استفاده از ماوس یا صفحه‌کلید انتخاب کند، که به‌طور پیش‌فرض امروز است.

۳-۳-۱. مفهوم سیگنال و اسلات

همان طور که می دانید اشیاء (آبجکت ها) در برنامه نویسی شئ گرا باید بتوانند با یکدیگر تعامل داشته باشند و پیغام هایی را با یکدیگر رد و بدل نمایند. یعنی، یک شئ بتواند برای شئ دیگری پیغامی ارسال کند. در این حالت شئ اول فرستاده شده و شئ دوم گیرنده است. سیگنال‌ها (signals) و اسلات‌ها (slots) وسیله تعامل بین اشیاء هستند. به عبارت دیگر سیگنال فروستنده و اسلات گیرنده می باشد. به عنوان مثال، فرض کنید که می خواهید با کلیک دکمه‌ای پنجره اصلی بسته شود. برای این منظور، سیگنال کلیک (()click) است و اسلات، تابعی است که پنجره اصلی را می بندد (یعنی، همان تابع ()close).

۴-۱. اضافه کردن ماژول PyQt به برنامه

برای استفاده از PyQt ابتدا باید بسته PyQt۵ را به برنامه اضافه نمود. برای این منظور، از دستور import به‌صورت زیر استفاده می‌شود:

```
import  PyQt5
```

انجام این عمل با دستور زیر نیز امکان‌پذیر است:

```
from PyQt5 import QtGui
```

۵-۱. اضافه کردن پنجره

پس از این‌که بسته PyQt۵ را به برنامه اضافه کردید، باید پنجره‌های موردنیازتان را ایجاد کنید تا بتوانید بر روی آن‌ها کنترل‌ها را قرار دهید. برای این منظور، مراحل زیر را انجام دهید:

۱. ماژول‌های موردنیاز برای ایجاد پنجره را به برنامه اضافه کنید (دستورات زیر را ببینید):

```
import sys
from PyQt5.QtWidgets import QApplication, Qdialog
```

دستور اول ماژول sys را به برنامه اضافه می‌کند تا بتوانید از sys.args و تابع ()sys.exit استفاده کنید و دستور دوم، کلاس‌های QApplication و QDialog از ماژول PyQt۵.QtWidgets را به برنامه اضافه می‌کند تا بتوانید از این کلاس‌ها در برنامه استفاده نمایید.

۲. تابعی برای ایجاد پنجره تعریف کرده و دستورات آن را به‌صورت زیر تایپ کنید:

```
def window():
   app = QApplication(sys.argv)
   win = QDialog()
   win.setGeometry(100,100,200,100)
```

```
    win.setWindowTitle("First window PyQt")
    win.show()
    sys.exit(app.exec_())
```

دستور اول، نمونه‌ای از کلاس QApplication به نام app ایجاد می‌کند، دستور دوم، نمونه‌ای از کلاس QDialog به نام win ایجاد می‌کند، دستور سوم، با متد ()setGeometry مکان نمایش پنجره، طول و عرض آن را به ترتیب نقطه (۱۰۰ ,۱۰۰)، ۲۰۰ و ۱۰۰ تعیین می‌کند. متد ()setGeometry به‌صورت زیر به کار می‌رود:

widget.setGeometry(xPosition, yPosition, Width, Height)

xPosition و yPosition، مختصات (x, y) قرار گرفتن ویجت را تعیین می‌کنند. اما، Width، عرض ویجت را مشخص می‌نماید و Height، ارتفاع ویجت را تعیین می‌کند.

دستور چهارم، با متد ()setWindowTitle عنوان پنجره را تعیین می‌کند. متد ()setWindowTitle به‌صورت زیر استفاده می‌شود:

windowWidget.setWindowTitle("text value")

"text value"، رشته‌ای است که می‌خواهید در عنوان پنجره نمایش داده شود.

دستور پنجم، پنجره ایجادشده را با استفاده از متد ()show نمایش می‌دهد. متد ()show به‌صورت زیر به کار می‌رود:

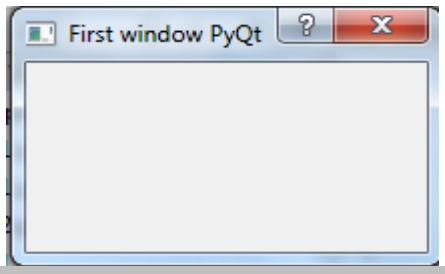
widget.show()

دستور ششم، وقتی پنجره بسته شد، با متد ()exit برنامه را خاتمه می‌دهد.

۳. در برنامه اصلی تابعی را که برای تعریف پنجره نوشته‌اید، فراخوانی کنید (با دستورات زیر):

```
if __name__ == '__main__':
    window()
```

با اجرای این دستور شکل زیر نمایش داده می‌شود و منتظر اجرای رویدادهای مختلف ماوس و صفحه‌کلید از کاربر می‌ماند:



مثال ۱–۱. برنامه‌ای که پنجره‌ای با عنوان "Fanavarienovin" در مختصات (۵۰ ,۲۰) با ارتفاع ۵۰ و عرض ۴۰۰ ایجاد می‌کند.

مراحل طراحی و اجرا

۱. پروژه جدیدی ایجاد کرده، دستورات آن را به‌صورت زیر تغییر دهید:

```
import sys
from PyQt5.QtWidgets import QApplication, QDialog
def window():
    app = QApplication(sys.argv)
    win = QDialog()
```

```
    win.setGeometry(20,50,400,50)
    win.setWindowTitle("Fanavarienovin")
    win.show()
    sys.exit(app.exec_())
if __name__ == '__main__':
    window()
```

دستورات اول و دوم، ماژول sys و کلاس‌های QApplication و QDialog را به برنامه اضافه می‌کنند. دستور سوم، تابعی به نام :()Window را تعریف می‌کند که اعمال زیر را انجام می‌دهد:

۱. از کلاس QApplication یک نمونه به نام app ایجاد می‌نماید.

۲. از کلاس QDialog یک نمونه به نام win ایجاد می‌کند.

۳. با متد ()setGeometry، پنجره win را به مکان (۵۰ ,۲۰) انتقال داده، طول و ارتفاع آن را به ترتیب ۴۰۰ و ۵۰ تعیین می‌نماید.

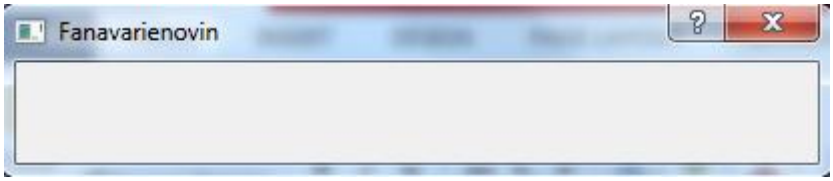
۴. با متد ()setWindowTitle، عنوان پنجره را Fanavarienovin تعیین می‌کند.

۵. با متد ()show پنجره را نمایش می‌دهد.

۶. برنامه را اجرا می‌کند، منتظر می‌ماند تا کاربر پنجره را ببندد. اکنون با متد ()exit برنامه را خاتمه می‌دهد.

در پایان، متد ()window را فراخوانی می‌کند.

۱. پروژه را ذخیره و اجرا کرده تا خروجی زیر را مشاهده نمایید:



۶-۱. ویجت QLabel

این ویجت برای نمایش متن‌ها، تصاویر ثابت و متحرک غیرقابل‌ویرایش توسط کاربر از قبیل عنوان فیلد، نمایش پیام‌ها و عکس‌ها به کار می‌رود. کلاس QLabel از کلاس QFrame مشتق می‌شود. برای استفاده از این کنترل باید مراحل زیر را انجام دهید:

۱. کلاس QLabel را به برنامه اضافه کنید (با دستور زیر):

```
from PyQt5.QtWidgets import Qlabel
```

۲. یک نمونه از کلاس QLabel بسازید (مانند دستور زیر):

```
lbl = QLabel()
```

۳. با متدهای کلاس QLabel، خواص نمونه ساخته‌شده را مقداردهی کنید(مانند دستورات زیر):

```
lbl.move(20, 10)
lbl.setText("Fanavarienovin")
lbl.adjustSize()
```

دستور اول، نمونه lbl را با متد ()move به نقطه‌ای با مختصات (۱۰ ,۲۰) انتقال می‌دهد. متد ()move به‌صورت زیر به کار می‌رود:

widget.move(xPosition, yPosition)

xPosition و yPosition مختصات نقطه‌ای است که ویجت widget باید به آن نقطه انتقال یابد.

دستور دوم، عنوان lbl را با متد ()setText به رشته "Fanavarienovin" تغییر می‌دهد. متد ()setText به‌صورت زیر به کار می‌رود:

widget.setText("text value")

"text value"، عنوان ویجت widget را مشخص می‌کند.

دستور سوم، اندازه lbl را از طریق متد ()adjustSize بر اساس محتوی کنترل تغییر می‌دهد. متد ()adjustSize به‌صورت زیر به کار می‌رود:

widget.adjustSize()

برخی از متدها و سیگنال‌های مهم کنترل QLabel عبارت‌اند از:

- **متد ()hide**، برای مخفی کردن ویجت به کار می‌رود و به‌صورت زیر استفاده می‌شود:

widget.hide()

- **متد ()show**، برای نمایش ویجت مخفی‌شده به کار می‌رود و به‌صورت زیر استفاده می‌شود:

widget.show()

- **متد ()clear**، محتوی ویجت را پاک می‌کند که به‌صورت زیر استفاده می‌شود:

widget.clear()

- **متد ()setAlignment**، برای تعیین چگونگی چینش متن در ویجت به کار می‌رود. این متد به‌صورت زیر استفاده می‌شود:

widget.setAlignment(alignment)

alignment، می‌تواند یکی از مقادیر Qt.AlignCenter (متن وسط‌چین می‌شود)، Qt.AlignLeft (متن چپ‌چین می‌شود) و Qt.AlignRight (متن راست‌چین می‌شود) را بپذیرد. برای استفاده از این مقادیر در آرگومان متد باید دستور زیر را به ابتدای برنامه اضافه کنید:

```
from PyQt5.QtCore import *
```

- **متد ()alignment**، مقدار alignment ویجت را برمی‌گرداند و به‌صورت زیر به کار می‌رود:

widget.alignment()

- **متد ()isLeftToRight**، مشخص می‌کند که آیا جهت نمایش اطلاعات در ویجت از چپ به راست است یا خیر؟ این متد به‌صورت زیر به کار می‌رود:

widget.isLeftToRight()

- **متد ()isRightToLeft**، مشخص می‌کند که آیا جهت نمایش اطلاعات در ویجت از راست به چپ است یا خیر؟ این متد به‌صورت زیر به کار می‌رود:

widget.isRightToLeft ()

- **متد ()isEnabled**، مشخص می‌کند که آیا ویجت فعال است یا خیر؟ این متد به‌صورت زیر به کار می‌رود:

widget.isEnabled()

- **متد ()isHidden**، مشخص می‌کند که آیا ویجت مخفی‌شده است یا خیر؟ این متد به‌صورت زیر به کار می‌رود:

widget.isHidden()

- **متد ()width**، عرض ویجت را برمی‌گرداند که به‌صورت زیر به کار می‌رود:

```
widget.width()
```

- **متد ()height**، ارتفاع ویجت را برمی‌گرداند که به‌صورت زیر به کار می‌رود:

```
widget.height()
```

- **متد ()geometry**، مختصات مکان هندسی، عرض و ارتفاع ویجت را برمی‌گرداند که به‌صورت زیر به کار می‌رود:

```
widget.gegometry()
```

- **متد ()x**، مختصات شروع ویجت روی محور x را برمی‌گرداند که به‌صورت زیر به کار می‌رود:

```
widget.x()
```

- **متد ()y**، مختصات شروع ویجت روی محور y را برمی‌گرداند که به‌صورت زیر به کار می‌رود:

```
widget.y()
```

- **متد ()hasSelectedText**، مشخص می‌کند که آیا بخشی از متن ویجت انتخاب‌شده است یا خیر؟ و به‌صورت زیر به کار می‌رود:

```
widget.hasSelectedText()
```

- **متد ()selectedText**، متن انتخاب‌شده ویجت را برمی‌گرداند که به‌صورت زیر به کار می‌رود:

```
widget.selectedText()
```

- **متد ()setFocus**، مکان‌نما را به ویجت موردنظر انتقال می‌دهد که به‌صورت زیر به کار می‌رود:

```
widget.setFocus()
```

- **متد ()setSelection**، بخشی از متن ویجت موردنظر را انتخاب می‌کند و به‌صورت زیر به کار می‌رود:

```
widget.setSelection(start, lenght)
```

start، اندیس شروع انتخاب را تعیین می‌کند که از صفر می‌باشد و lenght تعداد کاراکترهای که می‌خواهید انتخاب کنید، می‌باشد. برای استفاده از این متد ابتدا باید متد ()setFocus را بر روی ویجت موردنظر اجرا کنید.

- **متد ()setIndent**، نشانه متن برچسب را تنظیم می‌کند که به‌صورت زیر به کار می‌رود:

```
widget.setIntent(indent)
```

indent، عدد صحیحی است که نشانه متن برچسب را تعیین می‌کند.

- **متد ()setBuddy**، ویجت برچسب را به یک ویجت ورودی مرتبط می‌کند که به‌صورت زیر به کار می‌رود:

```
widget.setBuddy(buddy)
```

buddy، ویجتی است که ویجت ورودی مرتبط با ویجت برچسب را تعیین می‌کند.

- **متد ()setWordWrap**، تعیین می‌کند که اگر تعداد کاراکترهایی که در ویجت برچسب نمایش می‌شوند بیش از اندازه برچسب باشد، آیا نمایش متن در سطرهای بعدی ادامه یابد یا خیر؟ این متد به‌صورت زیر به کار می‌رود:

```
widget.setWordWrap (wordWrap)
```

wordWrap، مقداری منطقی است که تعیین می‌کند که آیا متن ویجت در سطر بعد ادامه یابد یا خیر؟

- **سیگنال linkActivated()**، اگر ویجت برچسب حاوی لینک تعبیه‌شده برای کلیک باشد، URL آن باز خواهد شد. برای انجام این عمل، خاصیت setOpenExternalLinks باید به True تنظیم شود.
- **سیگنال linkHovered**، وقتی متد اسلات مربوط به این سیگنال فراخوانی می‌شود که کاربر مکان‌نمای ماوس را روی برچسب با لینک تعبیه‌شده نگه‌داشته باشد.

مثال ۲–۱. برنامه‌ای که چند ویجت QLabel به پنجره اضافه می‌کند و متدهای مختلف را بر روی آن‌ها آزمایش می‌نماید.

مراحل طراحی و اجرا

۱. پروژه جدیدی ایجاد کرده، دستورات آن را به‌صورت زیر تغییر دهید:

```
import sys
from PyQt5.QtCore import *
from PyQt5.QtGui import *
from PyQt5.QtWidgets import QApplication, QWidget, QLabel,
   QVBoxLayout
def window():
    app = QApplication (sys.argv)
    win = QWidget ()
    l1=QLabel ()
    l2=QLabel ()
    l3=QLabel ()
    l4=QLabel ()
    l1.setText ("Hello Fanavarienovin")
    l2.move (40, 20)
    l4.setText ("<a href='www.fanavarienovin.net'>fanavarienovin</a>")
    l2.setText("<a href='#'>welcome to fanavarienovin site</a>")
    l1.setAlignment (Qt.AlignCenter)
    l3.setAlignment (Qt.AlignCenter)
    l4.setAlignment (Qt.AlignRight)
    l3.setPixmap (QPixmap ("g://python.jpg"))
    l1.setOpenExternalLinks (True)
    l4.linkActivated.connect (clicked)
    l2.linkHovered.connect (hovered)
    l1.setTextInteractionFlags (Qt.TextSelectableByMouse)
    vbox=QVBoxLayout()
    vbox.addWidget (l1)
    vbox.addStretch ()
    vbox.addWidget (l2)
    vbox.addStretch ()
    vbox.addWidget (l3)
    vbox.addStretch ()
    vbox.addWidget (l4)
    win.setLayout(vbox)
    win.setWindowTitle ("QLabel Test")
    win.show ()
    sys.exit (app.exec_ ())
def hovered():
        print( "hovering")
def clicked():
```

```
        print("clicked")
if __name__ == '__main__':
        window ()
```

این دستورات ابتدا ماژول‌ها و کلاس‌های موردنیاز را به برنامه اضافه می‌کنند، سپس توابع زیر را تعریف می‌کنند:

- **متد Window()** اعمال زیر را انجام می‌دهد:

۱. نمونه‌ای به نام app از کلاس QApplication ایجاد می‌کند.
۲. نمونه‌ای به نام win از کلاس QWidget ایجاد می‌کند.
۳. چهار نمونه به نام‌های l۱، l۲، l۳ و l۴ از کلاس QLabel() ایجاد می‌کند.
۴. با متد setText() عنوان l۱ را عبارت " Hello Fanavarienovin" تعیین می‌کند.
۵. با متد move() نمونه l۲ را به نقطه‌ای با مختصات (۲۰ ,۴۰) انتقال می‌دهد.
۶. با متد setText() عنوان l۴ را دستور HTML "<a href = 'www.fanavarienovin.net' " تعیین می‌کند.
۷. با متد setText() عنوان l۲ را دستور HTML " welcome to fanavarienovin site " تعیین می‌کند.
۸. با متد setAlignment() عنوان l۱ را وسط‌چین می‌کند.
۹. با متد setAlignment() عنوان l۳ را راست‌چین می‌کند.
۱۰. با متد setAlignment() عنوان l۴ را راست‌چین می‌کند.
۱۱. با متد setPixmap() تصویر python.jpg که در ریشه درایو G قرار دارد را در l۳ نمایش می‌دهد.
۱۲. با متد setOpenExternalLinks() مقدار خاصیت OpenExternalLinks نمونه l۲ را به True تغییر می‌دهد.
۱۳. اگر سیگنال LinkActivate برای l۴ اتفاق افتد، تابع Clicked() را فراخوانی می‌کند.
۱۴. اگر سیگنال LinkHovered برای l۲ اتفاق افتد، تابع hovered() را فراخوانی می‌کند.
۱۵. با متد setTextInteractionFlags() مقدار خاصیت TextInteractionFlags را Qt.TextSelectableMouse تعیین می‌کند.
۱۶. نمونه vbox را از کلاس QVBoxLayout ایجاد می‌کند.
۱۷. ویجت‌های l۱، l۲، l۳ و l۴ را به vbox اضافه می‌کند.
۱۸. با متد setLayout()، طرح‌بندی win را vbox تعیین می‌کند.
۱۹. با متد setWindowTitle() عنوان پنجره win را "QLabel test" تعیین می‌کند.
۲۰. با متد show() پنجره را نمایش می‌دهد.

۲۱. با متد exec_()، برنامه را اجرا می‌کند و منتظر می‌ماند تا کاربر پنجره را ببندد.

- **متد hovered()**، عبارت "hovering" را نمایش می‌دهد.
- **متد clicked()**، عبارت "clicked" را نمایش می‌دهد.

برنامه اصلی تابع Window() را فراخوانی می‌کند.

۲. پروژه را ذخیره و اجرا کرده تا خروجی زیر را مشاهده نمایید:

اکنون مکان‌نما را به welcome to fanavarienovin site انتقال داده و مکث کنید و سپس لینک fanavarienovin را کلیک کنید تا خروجی زیر را مشاهده کنید:

```
hovering
clicked
```

۷-۱. ویجت QLineEdit

این ویجت برای دریافت اطلاعاتی از قبیل مقادیر رشته‌ای، عددی که یک‌خطی باشند، به کار می‌رود.

برای استفاده از این ویجت باید مراحل زیر را انجام دهید:

۱. کلاس QLineEdit را به برنامه اضافه کنید (با دستور زیر):

```
from PyQt5.QtWidgets import QlineEdit
```

۲. یک نمونه از کلاس QLineEdit بسازید (مانند دستور زیر):

```
qle = QLineEdit()
```

۳. خواص نمونه ساخته‌شده را مقداردهی کنید (مانند دستورات زیر):

```
qle.move(20, 60)
```

```
qle.setText("")
qle.text()
```

دستور اول، نمونه qle را با متد ()move به نقطه‌ای با مختصات (۶۰ ,۲۰) انتقال می‌دهد. دستور دوم، محتوی qle را با متد ()setText به رشته خالی ("") تغییر می‌دهد. دستور سوم، محتوی qle را از طریق متد ()text برمی‌گرداند. متد ()text به‌صورت زیر به کار می‌رود:

widget.text()

برخی از متدهای مهم ویجت QLineEdit عبارت‌اند از:

- **متد ()setEchoMode**، مد نمایش متن در ویجت موردنظر را تعیین می‌کند و به‌صورت زیر به کار می‌رود:

widget.setEchoMode(echoMode)

echoMode، مد نمایش را مشخص می‌کند که می‌تواند یکی از مقادیر زیر را بپذیرد:

- **مقدار ۰ یا ثابت QLineEdit.Normal**، حالت نمایش نرمال ویجت QLineEdit را انتخاب می‌کند.
- **مقدار ۱ یا ثابت QLineEdit.NoEcho**، چیزی را در ویجت QLineEdit نمایش نمی‌دهد.
- **مقدار ۲ یا ثابت QLineEdit.Password**، ویجت QLineEdit را در حالت کلمه عبور نمایش می‌دهد.
- **مقدار ۳ یا ثابت QLineEdit.PasswordEchoOnEdit**، ویجت QLineEdit را در حالت کلمه عبور و نمایش روی تغییر قرار می‌دهد.
- **متد ()deselect**، متن انتخاب‌شده در ویجت موردنظر را از حالت انتخاب خارج می‌کند و به‌صورت زیر به کار می‌رود:

widget.deselect()

- **متد ()setMaxLength**، حداکثر تعداد کاراکتری که در ویجت موردنظر می‌توان تایپ نمود را تعیین می‌کند و به‌صورت زیر به کار می‌رود:

widget.setMaxLength(length)

length، عددی است که حداکثر تعداد کاراکترهایی می‌باشد که کاربر می‌تواند در ویجت **widget** وارد نماید.

- **متد ()maxLength**، حداکثر تعداد کاراکتری که می‌توان در ویجت موردنظر تایپ نمود را برمی‌گرداند و به‌صورت زیر به کار می‌رود:

widget.maxLength()

- **متد ()setFrame**، تعیین می‌کند که ویجت موردنظر فریم داشته باشد یا خیر؟ و به‌صورت زیر به کار می‌رود:

widget.setFrame(enabled)

اگر enabled دارای مقدار True باشد، ویجت **widget** فریم خواهد داشت، وگرنه(برای مقدار False)، این ویجت فریم ندارد.

- **متد ()hasFrame**، تعیین می‌کند که آیا ویجت موردنظر فریم دارد یا خیر؟ و به‌صورت زیر به کار می‌رود:

widget.hasFrame()

- **متد ()undo**، آخرین تغییر انجام‌شده در ویجت موردنظر را لغو می‌کند و به‌صورت زیر به کار می‌رود:

widget.undo()

- **متد ()redo**، آخرین فرمان undo شده در ویجت موردنظر را دوباره انجام می‌دهد و به‌صورت زیر به کار می‌رود:

widget.redo()

- **متد ()isUndoAvailable**، تعیین می‌کند که آیا undo قابل انجام است یا نه؟ و به‌صورت زیر به کار می‌رود:

widget.isUndoAvailable ()

- **متد ()isRedoAvailable**، تعیین می‌کند که آیا redo قابل انجام است یا نه؟ و به‌صورت زیر به کار می‌رود:

widget.isRedoAvailable ()

- **متد ()setCursorPosition**، موقعیت مکان‌نما را در ویجت تعیین می‌کند و به‌صورت زیر به کار می‌رود:

widget.setCursorPosition(position)

position، موقعیت مکان‌نما را مشخص می‌کند.

- **متد ()cursorPosition**، موقعیت مکان‌نما را در ویجت برمی‌گرداند و به‌صورت زیر به کار می‌رود:

widget.cursorPosition()

- **متد ()setReadOnly**، تعیین می‌کند که ویجت موردنظر فقط خواندنی باشد یا خیر؟ و به‌صورت زیر به کار می‌رود:

widget.setReadOnly(enable)

اگر **enable** دارای مقدار True باشد، ویجت **widget** فقط خواندنی است، وگرنه (برای مقدار False)، محتوی این ویجت توسط کاربر قابل‌تغییر است.

- **متد ()isReadOnly**، تعیین می‌کند که ویجت موردنظر فقط خواندنی است یا خیر؟ و به‌صورت زیر به کار می‌رود:

widget.isReadOnly()

- **متد ()pos**، موقعیت ویجت موردنظر را برمی‌گرداند و به‌صورت زیر به کار می‌رود:

widget.pos()

- **متد ()copy**، متن انتخاب‌شده در ویجت موردنظر را در حافظه موقت کپی می‌کند و به‌صورت زیر به کار می‌رود:

widget.copy()

- **متد ()cut**، متن انتخاب‌شده در ویجت موردنظر را به حافظه موقت انتقال می‌دهد و به‌صورت زیر به کار می‌رود:

widget.cut()

- **متد ()paste**، متن موجود در حافظه موقت را به مکان فعلی ویجت اضافه می‌کند و به‌صورت زیر به کار می‌رود:

widget.paste()

- **متد ()insert**، رشته‌ای را به انتهای ویجت موردنظر اضافه می‌کند و به‌صورت زیر به کار می‌رود:

widget.insert(insertString)

insertString، رشته‌ای است که می‌خواهید به انتهای ویجت widget اضافه شود.

- **متد selectAll()**،متن موجود در ویجت موردنظر را انتخاب می‌کند و به‌صورت زیر به کار می‌رود:

```
widget.selectAll()
```

- **متد setValidator()**، قوانین اعتبارسنجی را تنظیم می‌کند که به‌صورت زیر به کار می‌رود:

```
widget.setValidator(validator)
```

validator، از نوع Qvalidator است که می‌تواند یکی از مقادیر زیر باشد:

QIntValidator: ورودی را به عدد صحیح محدود می‌کند.

QDoubleValidator: بخش کسری عدد به ارقام مشخص‌شده محدود می‌شود.

QRegexpValidator: ورودی را در برابر عبارت Reggex بررسی می‌کند.

- **متد setInputMask()**، ماسک ترکیبی از کاراکترهای ورودی را بر روی ویجت اعمال می‌کند که به‌صورت زیر به کار می‌رود:

```
qle.setInputMask(inputmask)
```

inputmask، مقداری رشته‌ای است که نوع ماسک را تعیین می‌کند.

- **سیگنال cursorPositionChanged()**، وقتی رخ می‌دهد که مکان‌نمای ماوس حرکت کند.
- **سیگنال editingFinished()**، وقتی رخ می‌دهد که کاربر کلید "Enter" را فشار می‌دهد یا ویجت فوکوس را از دست می‌دهد.
- **سیگنال returnPressed()**، وقتی رخ می‌دهد که کاربر کلید "Enter" را فشار دهد.
- **سیگنال textChanged()**، وقتی رخ می‌دهد که متن ویجت تغییر می‌یابد.
- **سیگنال textEdited()**، وقتی رخ می‌دهد که متن ویجت ویرایش می‌شود.

مثال ۳-۱. برنامه‌ای که چند ویجت QLabel و چند ویجت QLineEdit به پنجره اضافه می‌کند، به‌طوری‌که متدهای مختلف ویجت QLineEdit را آزمایش می‌کند.

مراحل طراحی و اجرا

۱. پروژه جدیدی ایجاد کرده، دستورات آن را به‌صورت زیر تغییر دهید:

```
import sys
from PyQt5.QtCore import *
from PyQt5.QtGui import *
from PyQt5.QtWidgets import QApplication, QWidget, QLineEdit,
  QFormLayout
def window():
    app=QApplication (sys.argv)
    win=QWidget ()
    e1=QLineEdit ()
    e1.setValidator (QIntValidator ())
    e1.setMaxLength (4)
    e1.setAlignment (Qt.AlignRight)
    e1.setFont (QFont ("Arial", 20))
    e2=QLineEdit ()
    e2.setValidator (QDoubleValidator (0.99, 99.99, 2))
    flo=QFormLayout ()
    flo.addRow ("integer validator", e1)
    flo.addRow ("Double validator", e2)
    e3=QLineEdit ()
    e3.setInputMask ('+99_9999_999999')
```

```
    flo.addRow ("Input Mask", e3)
    e4=QLineEdit ()
    e4.textChanged.connect (textchanged)
    flo.addRow ("Text changed", e4)
    e5=QLineEdit ()
    e5.setEchoMode (QLineEdit.Password)
    flo.addRow ("Password", e5)
    e6=QLineEdit ("Hello Python")
    e6.setReadOnly (True)
    flo.addRow ("Read Only", e6)
    e5.editingFinished.connect (enterPress)
    win.setLayout (flo)
    win.setWindowTitle ("QLabel & QLineEdit test")
    win.show ()
    sys.exit (app.exec_ ())
def textchanged(text):
    print("contents of text box: " + text)
def enterPress():
    print("edited")
if __name__ == '__main__':
    window ()
```

این دستورات ابتدا ماژول‌ها و کلاس‌های موردنیاز را به برنامه اضافه می‌کنند، سپس توابع زیر را تعریف می‌کنند:

- **متد Window()** اعمال زیر را انجام می‌دهد:

۱. نمونه‌ای به نام app از کلاس QApplication ایجاد می‌کند.
۲. نمونه‌ای به نام win از کلاس QWidget ایجاد می‌کند.
۳. نمونه e۱ را از کلاس QLineEdit ایجاد می‌کند.
۴. با متد setValidator() ویجت e۱ را طوری تنظیم می‌کند تا فقط داده عددی صحیح را بپذیرد.
۵. با متد setMaxLength()، حداکثر تعداد کاراکترهای که در e۱ می‌توان وارد نمود را به ۴ تنظیم می‌کند.
۶. با متد setAlignment() متن e۱ را راست‌چین می‌کند.
۷. با متد setFont() فونت e۱ را "Arial" و اندازه آن را ۲۰ تعیین می‌کند.
۸. نمونه e۲ را از کلاس QLineEdit ایجاد می‌کند.
۹. با متد setValidator() اعتبارسنجی ویجت e۲ را طوری تنظیم می‌کند تا فقط عدد اعشاری با دقت مضاعف را بپذیرد.
۱۰. نمونه flo را از کلاس QFromLayout ایجاد می‌کند.
۱۱. در flo یک سطر ایجاد کرده، برچسب آن را integer validator و ویجت جلوی آن را e۱ تعیین می‌کند.

۱۲. در flo یک سطر دیگر ایجاد کرده، برچسب آن را Double validator و ویجت جلوی آن را e۲ تعیین می‌کند.
۱۳. نمونه e۳ را از کلاس QLineEdit ایجاد می‌کند.
۱۴. با متد ()setInputMask اعتبارسنجی ویجت e۳ را طوری تنظیم می‌کند تا رشته با فرمت '۹۹۹۹۹۹_۹۹۹_۹۹+' (یعنی، قالب مقادیر تلفن) را بپذیرد.
۱۵. یک سطر جدید به flo اضافه کرده، برچسب آن را Input Mask و ویجت جلوی آن را e۳ تعیین می‌کند.
۱۶. نمونه e۴ را از کلاس QLineEdit ایجاد می‌کند.
۱۷. سیگنال textChanged ویجت e۴ را طوری تنظیم می‌کند که اگر متن ویجت e۴ تغییر یابد، تابع ()textChanged را فراخوانی می‌کند.
۱۸. یک سطر جدید به flo اضافه کرده، برچسب آن را Text Changed و ویجت جلوی آن را e۴ تعیین می‌کند.
۱۹. نمونه e۵ را از کلاس QLineEdit ایجاد می‌کند.
۲۰. با متد ()setEchoMode ویجت e۵ را طوری تنظیم می‌کند تا متن واردشده در آن به‌صورت کلمه عبور نمایش داده شود.
۲۱. یک سطر جدید به flo اضافه کرده، برچسب آن را "Password" و متن جلوی آن را e۵ تعیین می‌کند.
۲۲. نمونه e۶ را از کلاس QLineEdit با مقدار Hello Python ایجاد می‌کند.
۲۳. با متد ()setReadOnly محتوی ویجت e۶ را به فقط خواندنی تنظیم می‌کند تا کاربر نتواند محتوی آن را تغییر دهد.
۲۴. یک سطر جدید به flo اضافه کرده، برچسب آن را "Read only" و متن جلوی آن را e۶ تعیین می‌کند.
۲۵. سیگنال editingFinished را برای ویجت e۵ طوری تنظیم می‌کند تا با رخ دادن این سیگنال تابع ()enterPress فراخوانی گردد.
۲۶. با متد ()setLayout، طرح‌بندی win را flo تعیین می‌کند.
۲۷. با متد ()setWindowTitle عنوان پنجره win را "QLabel & QLineEdit test" تعیین می‌کند.
۲۸. با متد ()show پنجره را نمایش می‌دهد.
۲۹. با متد ()__exec، برنامه را اجرا می‌کند و منتظر می‌ماند تا کاربر پنجره را ببندد.