

ТЕМА. PYQT

Мета. Вивчення основ розроблення графічного інтерфейсу користувача в середовищі PyQt.

Вступ. Бібліотека дозволяє розробляти застосунки з графічним інтерфейсом. Бібліотека PyQt не входить в склад стандартної бібліотеки Python, тому потрібно її встановити з репозиторію використовуваної ОС або з офіційного сайту *riverbankcomputing.com*. Прикладний інтерфус PyQt API є набір модулів, які включають велику кількість класів та функцій.

План.

1. Встановлення.
2. Виведення в PyQt стрічки "Hello world".
3. Основні класи PyQt.
4. Qt Designer.
5. Сигнали і слоти в PyQt.
6. Розміщення графічних елементів.
7. Базові графічні елементи.
8. Клас QDialog.
9. Діалог QMessageBox.
10. Одно- і багатодокументний інтерфейс.

1. Встановлення

Бібліотека PyQt сумісна з усіма популярними операційними системами, включно з Linux, Windows, and Mac OS.

Таблиця 1 – Версії PyQt4/PyQt5 для Windows

PyQt4-4.11.4-gpl-Py3.4-Qt4.8.7-x64.exe	Windows 64 bit installer
PyQt4-4.11.4-gpl-Py3.4-Qt4.8.7-x32.exe	Windows 32 bit installer
PyQt4-4.11.4-gpl-Py3.4-Qt5.5.0-x64.exe	Windows 64 bit installer
PyQt4-4.11.4-gpl-Py3.4-Qt5.5.0-x32.exe	Windows 32 bit installer
PyQt4-4.11.4-gpl-Py2.7-Qt4.8.7-x64.exe	Windows 64 bit installer
PyQt4-4.11.4-gpl-Py2.7-Qt4.8.7-x32.exe	Windows 32 bit installer
PyQt5-5.5-gpl-Py3.4-Qt5.5.0-x64.exe	Windows 64 bit installer
PyQt5-5.5-gpl-Py3.4-Qt5.5.0-x32.exe	Windows 32 bit installer

Встановлення PyQt в Linux Ubuntu з репозиторію:

```
>sudo apt-get install python-qt4
Або
sudo apt-get install pyqt5-dev-tools
```

або із інтернет сайту розробника на сторінці "завантаження":

PyQt-x11-gpl-4.11.4.tar.gz	Linux, UNIX source for PyQt4
----------------------------	------------------------------

2. Виведення в PyQt стрічки “Hello world”

Для створення простої GUI програми в PyQt потрібно виконати наступні кроки:

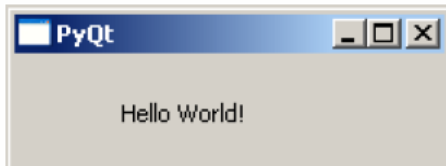
- Імпортувати QtGui модуль.
- Створити об’єкт програми.
- QWidget об’єкт створює основне вікно. Створити QLabel об’єкт.
- Встановити текст для виведення у вікно “hello world”.
- Визначити розмір і розміщення вікна за допомогою методу setGeometry.
- Ввести назву основного вікна.

```
import sys
from PyQt4 import QtGui

def window():
    app = QtGui.QApplication(sys.argv)
    w = QtGui.QWidget()
    b = QtGui.QLabel(w)
    b.setText("Hello World!")
    w.setGeometry(100,100,200,50)
    b.move(50,20)
    w.setWindowTitle("PyQt")
    w.show()
    sys.exit(app.exec_())

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

Результат виконання програми:



3. Основні модулі PyQt

PyQt має більш як 20 різних модулів. Нижче подані найчастіше використовувані:

Таблиця 2 – Модулі Qt

Бібліотека	Позначення в проектному файлі	Призначення
QtCore	core	Основний модуль, який складається з класів не зв'язаних з графічним інтерфейсом
QtGui	gui	Модуль для програмування графічного інтерфейсу
QtNetwork	network	Модуль для програмування мережі
QtOpenGL	opengl	Модуль для програмування графіки OGL
QtSql	sql	Модуль для програмування баз даних
QtSvg	svg	Модуль для роботи з SVG (Scalable Vector Graphics, масштабована векторна

		графіка)
QtXml	xml	Модуль підтримки XML, класи, які відносяться до SAX і DOM
QtScript	script	Модуль підтримки мови сценаріїв
Phonon	phonon	Модуль мультимедіа
QtWebKit	webkit	Модуль для створення Web-додатків
QtScriptTools	scripttools	Модуль підтримки додаткових можливостей мови сценарії (зневадник)
QtTest	test	Модуль, який містить класи для тестування коду
QtDesigner	qtdesigner	Класи розширення Qt Designer
QtAssistant	qtassistant	Онлайн-довідка

Клас QWidget() є фундаментальним для усіх віджетів і має наступних нащадків:

Таблиця 3 – Ієрархія класів віджетів

QWidget			
QDialog QPrintDialog QColorDialog QErrorMessage QFileDialog QFontDialog QInputDialog QMessageBox QPageSetupDialog QProgressDialog QWizard	QFrame QAbstractScrollArea QAbstractItemView QHeaderView QListView QListWidget QUndoView QTableView QTableWidget QTreeView QTreeWidget QColumnView	QAbstractSpinBox QDateTimeEdit QDateEdit QTimeEdit QDoubleSpinBox QSpinBox QComboBox QFontComboBox QAxWidget QCalendarWidget QColumnViewGrip	QLineEdit QMainWindow QMdiSubWindow QMenu QMenuBar QPrintPreviewWg QProgressBar QRubberBund QSizeGrip QSplashScreen QSplitterHandle QStatusBar
QAbstractButton QCheckBox QPushButton QRadioButton QToolButton	QGraphicsView QMdiAre QPlainTextEdit QScrollArea QTextEdit QTextBrowser	QDesignerAction QDesignerForm QDesignerObject QDesignerProperty QDesignerWidget	QSvgWidget QTabBar QTabWidget QToolBar QWSEmbeddedWg QWizardPage
QAbstractSlider QDial QScroolBar QSlider	QLCDNumber QLabel QSplitter QStackedWidget QToolBox	QDesktopWidget QDialogButtonBox QDockWidget QFocusFrame QGLWidget QGroupBox	QWorkSpace QWorkSpaceTitleBar QX11EmbedContainer QX11EmbedWidget

Нижче подані графічні елементи (віджети), які найчастіше використовуються.

№	Графічний елемент	Описання
1	QLabel	Використовується для відображення тексту або зображення
2	QLineEdit	Дозволяє користувачеві вводити один рядок тексту
3	QTextEdit	Дозволяє користувачеві вводити багаторядковий текст

4	QPushButton	Командна кнопка для виклику дії
5	QRadioButton	Дозволяє вибрати один з декількох варіантів
6	QCheckBox	Дозволяє вибрати більше ніж один варіант
7	QSpinBox	Дозволяє збільшити / зменшити ціле значення
8	QScrollBar	Дозволяє отримати доступ до вмісту віджета за межі діафрагми
9	QSlider	Дозволяє лінійно змінювати прив'язку.
10	QComboBox	Надає розкривний список елементів для вибору
11	QMenuBar	Горизонтальна панель з об'єктами QMenu
12	QStatusBar	Зазвичай в нижній частині QMainWindow надається інформація про стан
13	QToolBar	Зазвичай у верхній частині QMainWindow або плаваючою. Містить кнопки дій
14	QListView	Забезпечує вибір списку елементів у ListMode або IconMode
15	QPixmap	Представлення зображення поза екраном для відображення на об'єкті QLabel або QPushButton
16	QDialog	Модальне або модельне вікно, яке може повертати інформацію до батьківського вікна

Основне вікно застосунку на основі графічного інтерфейсу створюється за допомогою об'єкта віджета QMainWindow. В основному вікні за допомогою графічних елементів таб. 3 можуть створюватися інші графічні елементи, рис. 1:



Рисунок 1 – Основне вікно і можливі вбудовувані віджети

4. QT Designer

PyQt має інструмент для побудови графічного інтерфейсу, який називається Qt Designer. Qt Designer використовує метод перетягування графічних елементів з панелі інструментів на робоче поле основного вікна.

Створення графічного інтерфейсу за допомогою Qt Designer починається з вибору основного вікна програми, рис. 2.

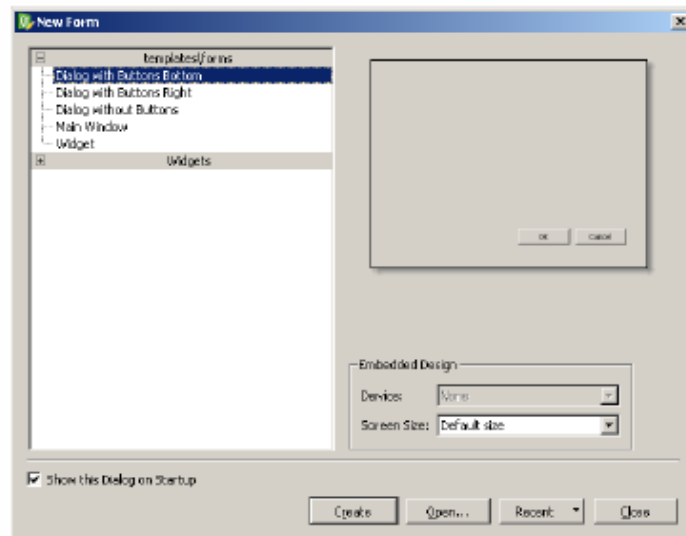


Рисунок 2 – Вибір основного вікна програми

Потім перетягуються необхідні віджети з лівої панелі інструментів. При цьому можна встановити необхідні значення властивостям віджета, рис. 3.

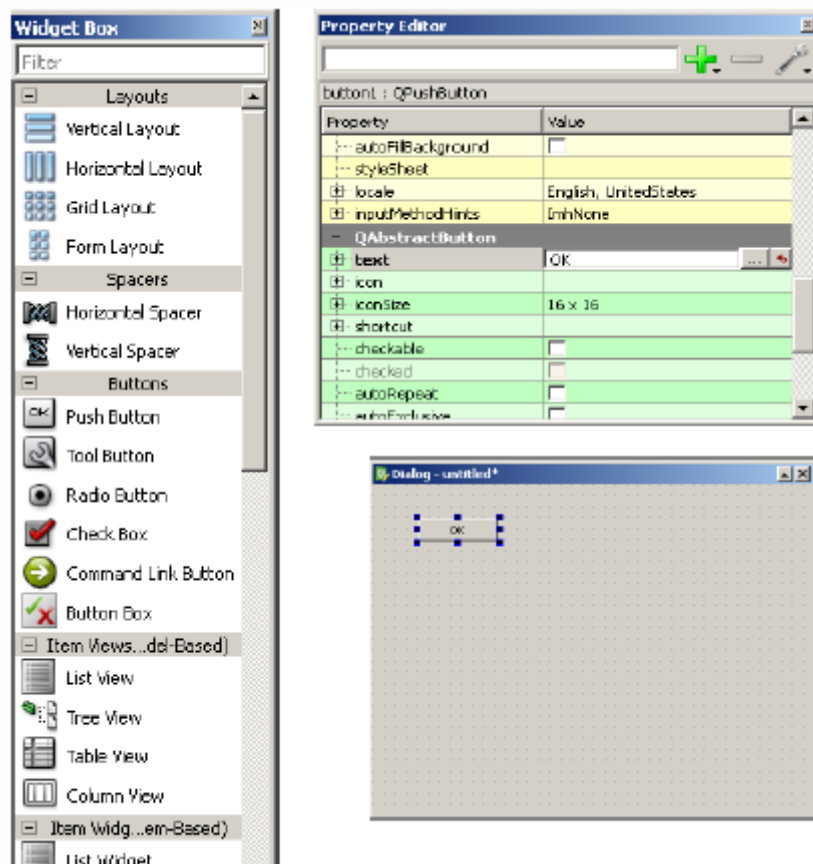


Рисунок 3 – Розміщення віджетів на основному вікні

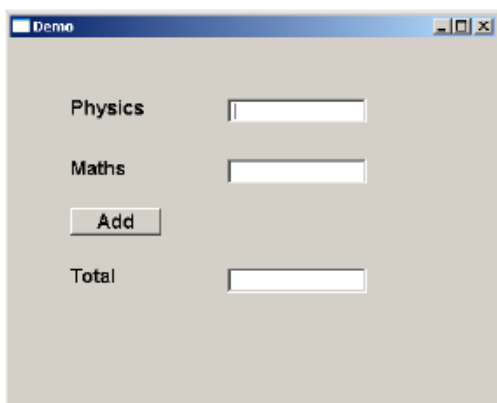
Розроблена форма графічного інтерфейсу зберігається у файлі із розширенням *.ui, наприклад, demo.ui. Цей файл містить XML-подання віджетів та їх властивостей. Для можливості виклику XML-подання форми з коду на мові Python використовується утиліта утиліта командного рядка `pyuic4`. Ця утиліта є обгорткою для модуля `uic`. Приклад використання `pyuic4`:

```
>pyuic4 -x demo.ui -o demo.py
```

Тут ключ `-x` додає генерацію додаткового коду до XML-подання форми, так що він стає самостійно виконуваним автономним застосунком:

```
if __name__ == "__main__":
    import sys
    app = QtGui.QApplication(sys.argv)
    Dialog = QtGui.QDialog()
    ui = Ui_Dialog()
    ui.setupUi(Dialog)
    Dialog.show()
    sys.exit(app.exec_())
```

Результат виконання програми `demo.py`:



Користувач може вводити дані у поля введення, але натискання кнопки `Add` не спричинить ніякої дії, натискання кнопки не пов'язане з будь-якою функцією. Реагування на відповідь, що генерується користувачем, називається обробкою подій. У PyQt передбачений спеціальний механізм оброблення подій, який реагує на дії користувача при роботі з графічними елементами.

5. Сигнали і слоти в PyQt

На відміну від програми консольного режиму, яка виконується послідовно, застосунок із графічним інтерфейсом є керованою подією. Функції або методи виконуються у відповідь на дії користувача, такі як натискання кнопки, вибір елементу з колекції або клацання миші тощо, це називається подіями.

При активації користувачем віджетів графічного інтерфейси вони стають джерелами подій. Кожен віджет PyQt, який виводиться з класу `QObject`, може відправляти (`emit`) "сигнал" у відповідь на одну або більше подій. Сигнал сам по собі не виконує жодних дій. Натомість він "з'єднаний" із "слотом". Слот може бути будь-якою функцією Python. У PyQt з'єднання між сигналом і слотом може бути досягнуто різними способами. Приклад найбільш часто використовуваного способу з'єднання:

```
QtCore.QObject.connect(widget, QtCore.SIGNAL('signalname'), slot_function)
```

Приклад коду:

```
import sys
from PyQt4.QtCore import *
from PyQt4.QtGui import *

def window():
    app = QApplication(sys.argv)
    win = QDialog()
    b1 = QPushButton(win)
    b1.setText("Button1")
    b1.move(50,20)
    b1.clicked.connect(b1_clicked)

    b2 = QPushButton(win)
    b2.setText("Button2")
    b2.move(50,50)
    QObject.connect(b2, SIGNAL("clicked()"), b2_clicked)

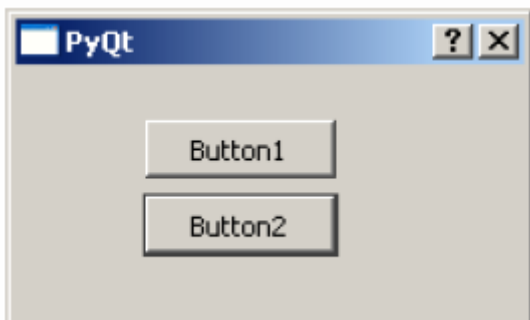
    win.setGeometry(100,100,200,100)
    win.setWindowTitle("PyQt")
    win.show()
    sys.exit(app.exec_())

def b1_clicked():
    print "Button 1 clicked"

def b2_clicked():
    print "Button 2 clicked"

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

Наведений вище код виводить основне вікно з двома кнопками. При натисканні на кнопки, виникають події, які обробляються функціями з друком повідомлень у консоль



Повідомлення у консолі:

```
Button 1 clicked
Button 2 clicked
```

6. Розміщення графічних елементів

Графічний елемент GUI може бути розміщений всередині вікна контейнера при заданні його абсолютних координат у пікселях. Координати задають розміщення вікна вказаного розміру за допомогою методу `setGeometry`.

```
QWidget.setGeometry(xpos, ypos, width, height)
```

У наступному фрагменті коду вікно верхнього рівня розміром 300×200 пікселів розміщується на моніторі в положенні 10, 10 пікселів (координати лівого верхнього кута).

```
import sys
from PyQt4 import QtGui

def window():
    app = QtGui.QApplication(sys.argv)
    w = QtGui.QWidget()
    b = QtGui.QPushButton(w)
    b.setText("Hello World!")
    b.move(50,20)

    w.setGeometry(10,10,300,200)
    w.setWindowTitle("PyQt")
    w.show()
    sys.exit(app.exec_())

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

У вікно додано графічний елемент кнопка (Push Button) з абсолютною координатою 50×20 пікселів відносно лівого верхнього кута основного вікна. Таке абсолютне позиціонування графічних елементів має ряд недоліків:

- позиція графічного елементу не змінюється при зміні розміру вікна;
- вигляд графічного елементу може різним на пристроях відображення з різною роздільною здатністю.
- складна модифікація макета форми, оскільки може знадобитися перепроєктування всієї форми.



а)



б)

Рисунок 4 – Розміщення кнопки: а) бажане; б) без зміщення при зміні розміру вікна

PyQt має засоби автоматичного горизонтального, вертикального і табличного розміщення графічних елементів.

Базовим для всієї групи менеджерів компоновання є клас `QLayout`. Це абстрактний клас успадкований від двох класів `QObject` і `QLayoutItem`.

Від класу `QLayout` успадковані класи `QGridLayout` і `QBoxLayout` (рис. 4.1). Клас `QGridLayout` керує табличним розміщенням, а від `QBoxLayout` успадковані два класи `QHBoxLayout` і `QVBoxLayout` для горизонтального і вертикального розміщення.

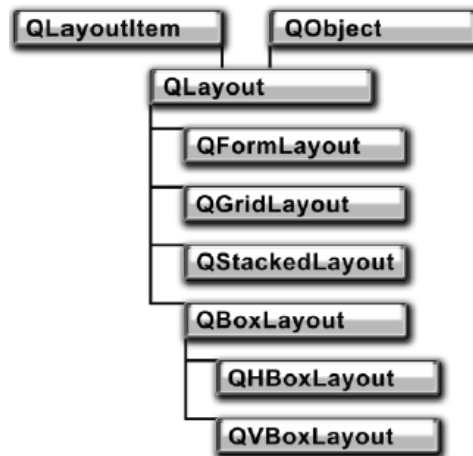


Рисунок 5 – Ієрархія класів менеджерів компоновання

7. Базові графічні елементи

№	Графічні елементи	Опис
1	QLabel	Об'єкт QLabel виконує роль заповнювача для відображення тексту надпису, зображення, фільму або анімованої GIF. Він також може використовуватися як мнемонічний ключ для інших віджетів.
2	QLineEdit	Об'єкт QLineEdit є полем введення. Він містить вікно, в якому можна ввести один рядок тексту. Для введення багаторядкового тексту призначений об'єкт QTextEdit.
3	QPushButton	Об'єкт QPushButton задає кнопку, яка при натисканні може реагувати викликом певної функції.
4	QRadioButton	Об'єкт QRadioButton задає набір радіо кнопок з надписами. Можна вибрати тільки одну кнопку з декількох. Клас похідний від класу QAbstractButton.
5	QCheckBox	Блок з набором прямокутників у яких можна задавати позначки.
6	QComboBox	Об'єкт QComboBox задає випадаючий список елементів для вибору. Вибраний елемент відображається в одному рядку.
7	QSpinBox	Об'єкт QSpinBox задає текстове поле і якому відображається ціле число за допомогою кнопки вгору/вниз, розташованої праворуч.
8	QSlider Widget & Signal	Об'єкт QSlider задає паз із повзунком, який дозволяє плавно міняти значення із обмеженого діапазону.
9	QMenuBar, QMenu & QAction	Об'єкт QMenuBar задає горизонтальну лінійку з елементами вибору (меню) безпосередньо під заголовком об'єкта QMainWindow.
10	QToolBar	Об'єкт QToolBar – це рухома панель інструментів, яка містить текстові кнопки, кнопк з іконками або інші віджети.
11	QInputDialog	Об'єкт QInputDialog – це діалогове вікно з текстовим полем і двома

		кнопками, OK і Cancel. У батьківському вікні появляються відповідні текстові повідомлення при натисканні користувачем кнопки Ok або клавіші Enter.
12	QFontDialog	Об'єкт QFontDialog – це діалогове вікно призначене для вибору шрифту.
13	QFileDialog	Цей віджет – це діалогове вікно призначене для вибору файлів. Вікно дозволяє користувачу переміщатися по файловій системі і вибирати файл для відкриття або збереження.
14	QTab	Якщо форма має занадто багато полів для відображення одночасно, їх можна розташувати на різних сторінках, розташованих під кожною вкладкою віджета-вкладки. Об'єкт QTabWidget надає панель вкладок і сторінок.
15	QStacked	Об'єкт допомагає в ефективному використанні клієнтської області вікна.
16	QSplitter	Об'єкт QSplitter дозволяє розділити основне вікно на декілька підвікон.
17	QDock	Об'єкт QDoc задає прикріплюване підвікно, яке може залишатися у плаваючому стані або може бути приєднане до основного вікна у заданій позиції.
18	QStatusBar	Об'єкт QMainWindow задає горизонтальну смугу внизу основного вікна як рядок стану. В ній відображується інформація про постійний або контекстний статус.
19	QList	Об'єкт QListWidget підтримує додавання і видалення елементів із списку. Кожен елемент списку є об'єктом QListWidgetItem. ListWidget може бути встановлений як багатовимірний.
20	QScrollBar	Об'єкт QScrollBar задає смугу прокрутки, яка дозволяє отримати доступ до частини документа, який знаходиться за межами видимої області. Таке він забезпечує візуальний індикатор поточної позиції.
21	QCalendar	Об'єкт QCalendar підтримує. Користувач може вибрати поточну дату за допомогою миші або клавіатури.

8. Клас QDialog

Об'єкт QDialog подає вікно верхнього рівня, яке обробляє відповіді користувача. Об'єкт можна налаштувати на режим діалогу Modal (блокування батьківського вікна до отримання відповіді з вікна діалогу) або режим Modeless (без блокування батьківського вікна). В PyQt є декілька попередньо налаштованих діалогів, таких як InputDialog, FileDialog, FontDialog і т.п.

В наступному прикладі, атрибут WindowModality вікна діалогу визначає роботу батьківського вікна з блокуванням або без блокування. Будь-яку одну кнопку в діалоговому вікні можна встановити у значення за замовчуванням. Діалог завершується методом `QDialog.reject()`, коли користувач натискає клавішу Escape.

При натисканні кнопки `PushButton` у вікні верхнього рівня `QWidget`, створюється вікно діалогу. Діалогове вікно у змусі заголовку не має керуючих елементів для його згортання або розгортання.

Користувач не може перевести діалогове вікно у фоновий режим, оскільки його атрибут `WindowModality` встановлено в режим `ApplicationModal`.

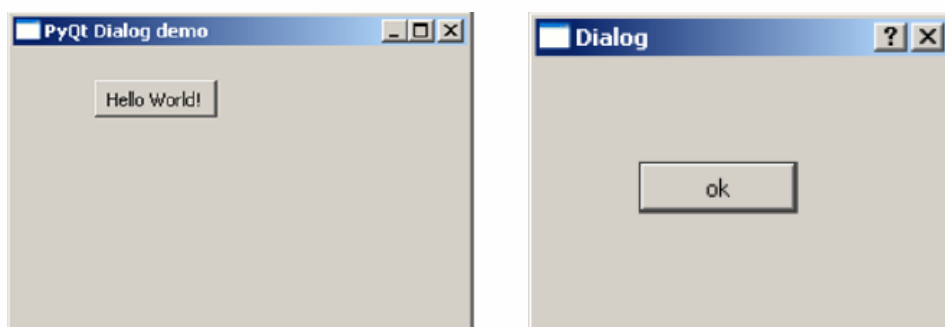
```
import sys
from PyQt4.QtGui import *
from PyQt4.QtCore import *

def window():
    app = QApplication(sys.argv)
    w = QWidget()
    b = QPushButton(w)
    b.setText("Hello World!")
    b.move(50,50)
    b.clicked.connect(showdialog)
    w.setWindowTitle("PyQt Dialog demo")
    w.show()
    sys.exit(app.exec_())

def showdialog():
    d = QDialog()
    b1 = QPushButton("ok",d)
    b1.move(50,50)
    d.setWindowTitle("Dialog")
    d.setWindowModality(Qt.ApplicationModal)
    d.exec_()

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

Результат виконання програми:







9. Діалог `QMessageBox`

`QMessageBox` є стандартним модальним діалогом для висвітлення інформаційних повідомлень і загальних запитань у відповідь на клацання користувачем на одну із стандартних кнопок.

Важливі методи і повідомлення асоційовані із класом `QMessage` показані у наступній таблиці.

Таблиця 3 – Методи і повідомлення класу `QMessageBox`

№	Методи та опис
---	----------------

1	<p>setIcon() – висвічує попередньо визначену піктограму, що відповідає серйозності повідомлення</p> <div>  Question </div> <div>  Information </div> <div>  Warning </div> <div>  Critical </div>
2	setText() – встановлює текст основного повідомлення для відображення
3	setInformativeText() – відображає додаткову інформацію
4	setDetailText() – відображає кнопку Details. Цей текст з'являється після клацання на кнопці
5	setTitle() – відображає назву діалогового вікна задану користувачем
6	<p>setStandardButtons()</p> <p>Список стандартних кнопок для відображення. Кожна кнопка асоціюється з</p> <p>QMessageBox.Ok 0x00000400</p> <p>QMessageBox.Open 0x00002000</p> <p>QMessageBox.Save 0x00000800</p> <p>QMessageBox.Cancel 0x00400000</p> <p>QMessageBox.Close 0x00200000</p> <p>QMessageBox.Yes 0x00004000</p> <p>QMessageBox.No 0x00010000</p> <p>QMessageBox.Abort 0x00040000</p> <p>QMessageBox.Retry 0x00080000</p> <p>QMessageBox.Ignore 0x00100000</p>
7	setDefaultButton() – встановлює кнопку за замовчуванням. При натисканні клавіші Enter кнопка видає сигнал клікання
8	setEscapeButton() – встановлює кнопку, яку слід обробити як клікнуту при натисненні клавіші Escape.

У наступному прикладі для сигналу клікання на кнопці, підключена функція відобразить вікно повідомлення діалогу.

```
msg = QMessageBox()
msg.setIcon(QMessageBox.Information)
msg.setText("This is a message box")
msg.setInformativeText("This is additional information")
msg.setWindowTitle("MessageBox demo")
msg.setDetailedText("The details are as follows:")
```

Функція setStandardButton() відображає потрібні кнопки.

```
msg.setStandardButtons(QMessageBox.Ok | QMessageBox.Cancel)
```

Сигнал `buttonClicked()`, підключений до функції слота, яка ідентифікує заголовок джерела сигналу.

```
msg.buttonClicked.connect(msgbtn)
```

Приклад програми діалогу.

```
import sys
from PyQt4.QtGui import *
from PyQt4.QtCore import *

def window():
    app = QApplication(sys.argv)
    w = QWidget()
    b = QPushButton(w)
    b.setText("Show message!")

    b.move(50,50)
    b.clicked.connect(showdialog)
    w.setWindowTitle("PyQt Dialog demo")
    w.show()
    sys.exit(app.exec_())

def showdialog():
    msg = QMessageBox()
    msg.setIcon(QMessageBox.Information)

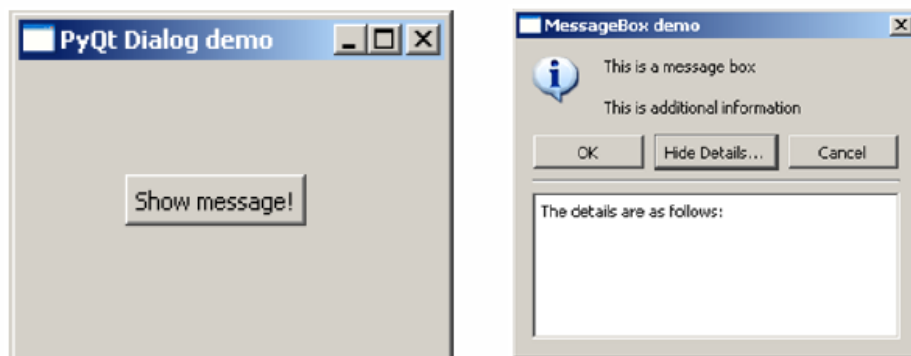
    msg.setText("This is a message box")
    msg.setInformativeText("This is additional information")
    msg.setWindowTitle("MessageBox demo")
    msg.setDetailedText("The details are as follows:")
    msg.setStandardButtons(QMessageBox.Ok | QMessageBox.Cancel)
    msg.buttonClicked.connect(msgbtn)

    retval = msg.exec_()
    print "value of pressed message box button:", retval

def msgbtn(i):
    print "Button pressed is:", i.text()

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

Результат виконання:



10. Одно- і багатодокументний інтерфейс

Типовий за стосунок із графічним інтерфейсом може мати декілька вікон. Графічні елементи із вкладками і стеками дозволяють одночасно активувати одне з таких вікон. Однак в багатьох випадках цей підхід не підходить, так як інші вікна є приховані.

Один із способів одночасного відображення декількох вікон полягає у створенні їх як незалежних вікон. Такий за стосунок називається однодокументним (SDI – SingleDocumentInterface). Для цього потрібно більше ресурсів пам'яті, оскільки кожне вікно може мати власну систему меню, панель інструментів тощо.

Багатодокументні застосунки (MDI – MultipleDocumentInterface) споживають менші ресурсів пам'яті. Підвікна вкладаються всередину основного контейнера по відношенню один до одного. Віджет контейнера називається `QMdiArea`.

Віджет `QMdiArea` звичайно заміщує центральний віджет об'єкта `QMainWindow`. Дочірні вікна в цій області є екземпляри класу `QMdiSubWindow`. Можна встановити будь-який `QWidget` як внутрішній віджет об'єкта `subWindow`. Підвікна в області MDI можуть бути розташовані у каскадному або плитковому стилі.

У наступній таблиці подані важливі методи класу `QMdiArea` і класу `QMdiSubWindow`:

Таблиця 3 – Методи класу `QMdiArea` і класу `QMdiSubWindow`

№	Графічний елемент	Опис
1	<code>addSubWindow()</code>	Додає віджет як нове підвікно в області MDI
2	<code>removeSubWindow()</code>	Вилучає віджет, який є внутрішнім віджетом підвікна
3	<code>setActiveSubWindow()</code>	Активує підвікно
4	<code>cascadeSubWindows()</code>	Розміщує підвікна в MDiArea каскадом
5	<code>tileSubWindows()</code>	Розміщує підвікна в MDiArea плитками
6	<code>closeActiveSubWindow()</code>	Закриває активне підвікно
7	<code>subWindowList()</code>	Повертає список підвікон в області MDI
8	<code>setWidget()</code>	Встановлює <code>QWidget</code> як внутрішній віджет екземпляра <code>QMdiSubwindow</code>

Об'єкт `QMdiArea` випромінює сигнал `subWindowActivated()` коли сигнал `windowStateChange()` випромінюється об'єктом `QMdiSubWindow`.

У наступному прикладі вікно верхнього рівня утворене з `QMainWindow` має меню і `MdiArea`.

```
self.mdi = QMdiArea()
self.setCentralWidget(self.mdi)
bar = self.menuBar()
file = bar.addMenu("File")

file.addAction("New")
file.addAction("cascade")
file.addAction("Tiled")
```

Сигнал меню `Triggered()` підключений до функції `windowaction()`.

```
file.triggered[QAction].connect(self.windowaction
```

Нова дія меню, додає підвікно в область MDI з назвою, що містить інкрементний номер.

```
MainWindow.count = MainWindow.count+1
sub = QMdiSubWindow()
sub.setWidget(QTextEdit())
sub.setWindowTitle("subwindow"+str(MainWindow.count))
self.mdi.addSubWindow(sub)
sub.show()
```

Каскадні та плиткові кнопки меню впорядковують поточно відображені підвікна каскадним і плитковим способом відповідно.

Приклад програми розміщення вікон каскадним і плитковим способом.

```
import sys
from PyQt4.QtCore import *
from PyQt4.QtGui import *

class MainWindow(QMainWindow):
    count = 0

    def __init__(self, parent = None):
        super(MainWindow, self).__init__(parent)
        self.mdi = QMdiArea()
        self.setCentralWidget(self.mdi)
        bar = self.menuBar()

        file = bar.addMenu("File")
        file.addAction("New")
        file.addAction("cascade")
        file.addAction("Tiled")
        file.triggered[QAction].connect(self.windowaction)
        self.setWindowTitle("MDI demo")

    def windowaction(self, q):
        print "triggered"

    if q.text() == "New":
        MainWindow.count = MainWindow.count+1
        sub = QMdiSubWindow()
        sub.setWidget(QTextEdit())
        sub.setWindowTitle("subwindow"+str(MainWindow.count))
        self.mdi.addSubWindow(sub)
        sub.show()

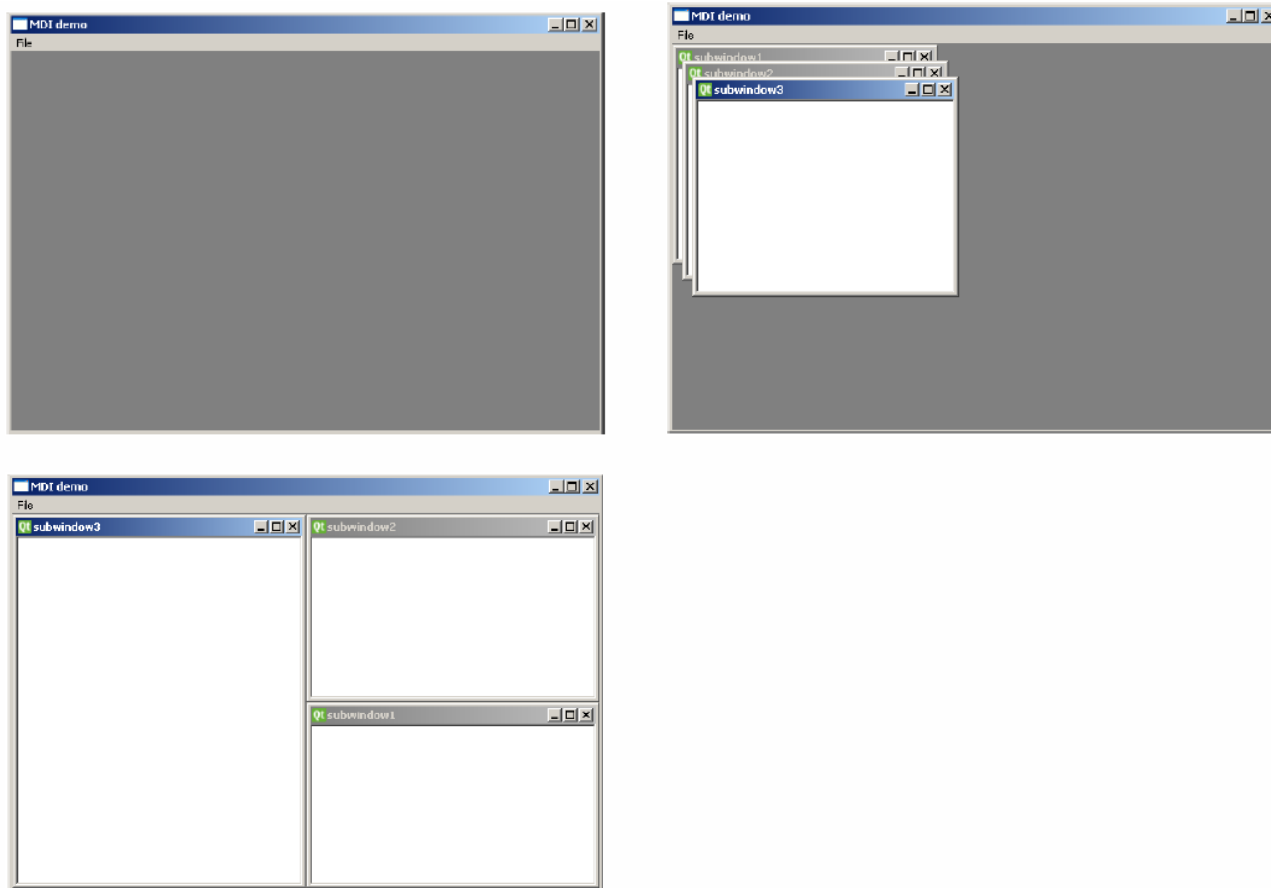
    if q.text() == "cascade":
        self.mdi.cascadeSubWindows()

    if q.text() == "Tiled":
        self.mdi.tileSubWindows()

    def main():
        app = QApplication(sys.argv)
        ex = MainWindow()
        ex.show()
        sys.exit(app.exec_())

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

Результат виконання:



Список літератури

1. Гордеев А.В., Молчанов А.Ю. Системное программное обеспечение. СПб.: ДМК. – 2003. – 736.
2. Ноа Гифт, Джереми М. Джонс. Python в системном администрировании. СПб.: Символ-Плюс, 2009. – 512 с.
3. Бизли Д. Python. Подробный справочник. СПб.: Символ-Плюс, 2010, 864 с.
4. Бизли Д.М. Язык программирования Python. Справочник. – Диасофт, 2000.
5. Тейнсли Д. Язык Shell Linux и Unix. – Пер. с англ. – СПб.: БХВ-Петербург, 2001. – 512с.
6. Феникс Т., Шварц Р. Изучаем Perl. – Пер. с англ. – СПб.: БХВ-Петербург, 2002. – 288 с.
7. Лутц М. Программирование на Python. – Пер. с англ. – СПб.: Символ-Плюс, 2002. – 1136с.
8. Брент Б. Уелш, Кен Джонс, Джеффри Хоббс. Практическое программирование на Tcl и Tk. – Пер. с англ. – СПб.: Изд. дом “Вильямс”, 2004. – 1138 с.
9. Сузи Р.А. Язык программирования Python. – Бином. М.: - 2006.
10. Фитцджеральд М. Изучаем Ruby. – Пер. с англ. – СПб.: БХВ-Петербург, 2008. – 336 с.
11. Mark Summerfield. Rapid GUI programming with Python and Qt. – NJ: Prentice Hall, 2008. – 625 с.
12. Саммерфилд М. Программирование на Python 3. Подробное руководство. – Пер. С англ.. – СПб.: Символ-Плюс, 2009. – 608 с.
13. Бизли Д. Python. Подробный справочник. – Пер. с англ. – СПб.: Символ-Плюс, 2010. – 864 с., ил.
14. Хахаев И. А. Практикум по алгоритмизации и программированию на Python: / И.А.Хахаев М. : Альт Линукс, 2010. – 126 с.
15. Лутц М. Программирование на Python, том I, 4-е издание. – Пер. с англ. – СПб.: Символ-Плюс, 2011. – 992 с., ил.

16. Лутц М. Программирование на Python, том II, 4-е издание. – Пер. с англ. – СПб.: Символ-Плюс, 2011. – 992 с., ил.
17. Прохорецок Н. А. Python 3 и PyQt. Разработка приложений. – СПб.: БХВ-Петербург, 2012. – 704 с.
18. Марк Саммерфилдт. Python на практике / Пер. с англ. Слинкин А.А. – М.: ДМК Прес, 2014. – 338 с.
19. Чан, Уэсли. Python: создание приложений. Библиотека профессионала, 3-е изд. Пер. с англ. - М. : ОО "И.Д. Вильяме", 2015. - 816 с.
20. <https://www.riverbankcomputing.com/news/pyqt-5112>
21. <https://www.tutorialspoint.com/pyqt/>

Запитання.

1. Призначення основних модулів PyQT.
2. Базові віджети PyQt.
3. Призначення і можливості Qt Designer.
4. Призначення сигналів та слотів в Qt
5. Переваги і недоліки способів розміщення графічних елементів.
6. Модальні і немодальні діалоги.
7. Методи і повідомлення класу QMessageBox.
8. Одно- і багато віконні інтерфейси.

Завдання.

1. Написати програму, яка виводить у вікні ім'я та прізвище та виводить привітання
2. Створити у Qt Designer форму з полем для введення даних та кнопкою, при натисканні якої виводиться надпис із введенними даними.
3. Написати програму для введення даних у поля з назвами “Студент”, “Дисципліна”, “Оцінка” і запису значень у файл.
4. Написати програму для створення радіо кнопок і встановлення їх у стан, зчитаний із файлу.
5. Написати програму для створення списку вибору, значення списку зчитується із файлу.
6. Написати програму для створення графічного елементу QSpinBox, значення якого виводиться у текстовке поле.
7. Написати програму для створення форми із смугою меню.
8. Написати програму для створення діалогу QFileDialog і копіювання вибраного файлу у каталог tmp.
9. Написати програму з використанням класу QMessageBox для висвітлення інформаційного повідомлення у відповідь на клацання користувачем на одну із стандартних кнопок.
10. Написати програму, яка створює однодокументний інтерфейс для двох вікон.
11. Написати програму, яка створює багатодокументний інтерфейс для двох вікон і розміщує їх каскадним і плитковим способом.